Citation for published version:
Bulhões , T, Subramanian, A, Erdoan, G & Laporte, G 2018, 'The static bike relocation problem with multiple vehicles and visits', *European Journal of Operational Research*, vol. 264, no. 2, pp. 508-523. https://doi.org/10.1016/j.ejor.2017.06.028

DOI:
10.1016/j.ejor.2017.06.028

Publication date:
2018

Document Version
Peer reviewed version

Link to publication

Publisher Rights
CC BY-NC-ND

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

The static bike relocation problem with multiple vehicles and visits

Teobaldo Bulhões[a], Anand Subramanian[b,*], Güneş Erdoğan[c], Gilbert Laporte[d]

[a] *Instituto de Computação, Universidade Federal Fluminense*
Rua Passo da Pátria 156, São Domingos, 24210-240, Niterói–RJ, Brazil.
[b] *Departamento de Sistemas de Computação, Centro de Informática, Universidade Federal da Paraíba*
Rua dos Escoteiros, Mangabeira, 58058-600, João Pessoa–PB, Brazil.
[c] *School of Management, University of Bath, Bath, BA2 7AY, UK.*
[d] *Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, Canada H3T 2A7.*

Abstract

This paper introduces the *static bike relocation problem with multiple vehicles and visits*, the objective of which is to rebalance at minimum cost the stations of a bike sharing system using a fleet of vehicles. The vehicles have identical capacities and service time limits, and are allowed to visit the stations multiple times. We present an integer programming formulation, implemented under a branch-and-cut scheme, in addition to an iterated local search metaheuristic that employs efficient move evaluation procedures. Results of computational experiments on instances ranging from 10 to 200 vertices are provided and analyzed. We also examine the impact of the vehicle capacity and of the number of visits and vehicles on the performance of the proposed algorithms.

Keywords: Routing, Shared mobility systems, Bike sharing, Pickup and delivery

1. Introduction

We study the problem of rebalancing at minimum cost the stations of a *bike sharing system* (BSS) by relocating the bikes using a fleet of vehicles. The inventories of the stations and the travel times between stations are assumed to be static during the rebalancing operation. The fleet is composed of identical vehicles with a given capacity and service time limit, which depart from and return to the *depot*. The cost of the operation is measured as the total travel time of the fleet. The vehicles can visit the statios multiple times, up to a given limit, but a station can only be served by one vehicle of the fleet. In addition to the

*Corresponding author
Email addresses: tbulhoes@ic.uff.br (Teobaldo Bulhões), anand@ci.ufpb.br, anandsubraman@gmail.com (Anand Subramanian), g.erdogan@bath.ac.uk (Güneş Erdoğan), gilbert.laporte@cirrelt.ca (Gilbert Laporte)

travel times, we include the handling times of the bikes within the service time limit of the vehicles to ensure that the workload constraint is not violated. This problem is called the *static bicycle relocation problem with multiple vehicles and visits* (SBRP-MVV).

There are now more than 7000 BSSs in the world, as stated in the recent survey by Laporte et al. (2015), and this number is growing at an increasing rate. There exists a rich body of research on the problem of rebalancing BSSs, mainly for two variants of the problem: the *static* version and the *dynamic* version. The main difference between the two variants is the customer demand during the rebalancing operation, which is assumed to be zero for the static variant while it can be non-zero for the dynamic variant. We refer the interested reader to the papers by Nair and Miller-Hooks (2011), Contardo et al. (2012), and Chemla et al. (2013a) for further details of the dynamic version. In what follows, we provide a brief review of the literature on the static version.

The studies on the static version can be traced back to the seminal paper of Benchimol et al. (2011) in which the authors introduced the *static stations balancing problem* (SSBP) and proved it to be $\mathcal{NP}$-Hard. The SSBP aims at finding a minimum cost route for a single vehicle that can visit the same station multiple times. The vehicle can drop bikes temporarily at intermediate locations for future pickup, i.e. *preemption* was allowed. The assumption of a single vehicle was deemed to be realistic due to the size of the BSS at the time, and since the problem would arise as a subproblem for the BSS with multiple vehicles.

The studies that followed introduced variants by changing three features: the number of vehicles, the number of visits to stations, and the preemption property. Notably, a few studies have incorporated additional features such as minimizing user dissatisfaction (Raviv et al., 2013), demand intervals for the stations (Erdoğan et al., 2014), and multiple types of bikes (Li et al., 2016). The reach of the state-of-the-art exact algorithms is 60 stations across the variants of the static bike relocation problem. The fleet size depends on the instances being studied, and is in the range of [1–20] with and average of 3.1 for the real-world instances of Dell'Amico et al. (2016). Table 1 provides a summary of the literature on the static bike relocation problems.

In this paper, we study the SBRP-MVV, which is a realistic representation of the real-world problem. The necessity of multiple visits arises from the existence of large, central bike stations the demand of which cannot be satisfied in a single visit. We present an iterated local search heuristic for the SBRP-MVV that benefits from subsequence based data structures, allowing the algorithm to perform move evaluations in amortized constant time. We also develop and present an integer linear programming formulation and an

Table 1: Literature on static bike relocation problems

		Single vehicle	**Multiple vehicles**
Single visit	Nonpreemptive	Erdoğan et al. (2014) † Ho and Szeto (2014) † ‡ Li et al. (2016) † ‡	Lin and Chou (2012) ‡ Raviv et al. (2013) † Dell'Amico et al. (2014) † Forma et al. (2015) ‡ Dell'Amico et al. (2016) † ‡
Multiple visits	Nonpreemptive	Erdoğan et al. (2015) †	Gaspero et al. (2013) ‡ Espegren et al. (2016) †
	Preemptive	Benchimol et al. (2011) † Chemla et al. (2013b) ‡ Erdoğan et al. (2015) † Cruz et al. (2017) ‡	Rainer-Harbach et al. (2015) ‡ Alvarez-Valdes et al. (2016) ‡

† *Exact algorithm*
‡ *Heuristic algorithm*

associated branch-and-cut algorithm for the SBRP-MVV.

The remainder of the paper is organized as follows. Section 2 formally defines the SBRP-MVV and presents an integer programing formulation. Section 3 describes the branch-and-cut algorithm. Section 4 explains the proposed iterated local search metaheuristic. Section 5 contains the computational experiments. Finally, Section 6 presents the concluding remarks of this work.

2. Mathematical formulation

The SBRP-MVV can be formally defined as follows. We are given a complete and directed graph $G = (V, A)$, where V is the vertex set and A is the arc set. Vertex 0 is the depot, while the remaining ones are the bike stations. Each station $i \in V \setminus \{0\}$ has a pickup or a delivery demand q_i. We assume that $q_i > 0$ indicates a pickup demand, whereas $q_i < 0$ denotes a delivery demand. We also assume that $q_0 = -\sum_{j \in V \setminus \{0\}} q_j$. Each arc $a \in A$ has an associated travel time c_a. A fleet of K identical vehicles with capacity Q is available at the depot. A route is a sequence of vertices starting and ending at the depot, each vertex corresponding to a visit to a station, with a pickup or delivery amount associated with each visit. A route duration limit L is imposed for each route. There is a service time proportional to the number of bikes to be delivered or collected at a station. More precisely, this service time is given by the product of the handling time h and the number of bikes delivered or collected. Moreover, a station is allowed to be visited at most N times by the same vehicle, an assumption that enables us to model the problem effectively. We assume that a station cannot be visited by different vehicles, i.e., it cannot appear in two

distinct routes in a feasible solution. Finally, to ensure feasibility we impose that $|q_i| \leq NQ$ and $q_0 \leq KQ$. The objective is to minimize the total travel time.

We define the network $G' = (V', A')$, where $V' = V_C \cup \{0, d\}$. Set V_C represents visits to the stations, and contains N nodes for each station. Vertices 0 and d are the starting and ending points, respectively, for all routes. The arc set A' contains all possible arcs, except those between nodes representing visits to the same station, and therefore there is no arc between 0 and d. Moreover, there is no arc leaving node d or entering node 0.

Figure 1 depicts an example of a network G' involving two stations and $N = 2$.

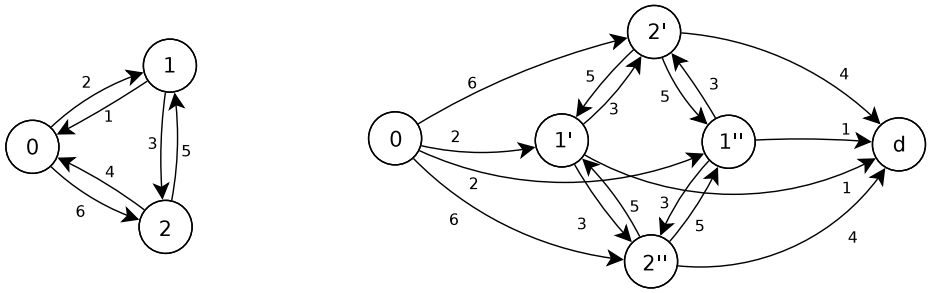


Figure 1: Example of a network G' involving two stations

We define the following notation:

- $\delta^+_{(i)} \subseteq A'$: the set of arcs leaving vertex $i \in V'$;
- $\delta^-_{(i)} \subseteq A'$: the set of arcs entering vertex $i \in V'$;
- $\delta^+_{(S)} \subseteq A'$: the set of arcs leaving set $S \subseteq V'$;
- $\delta^-_{(S)} \subseteq A'$: the set of arcs entering set $S \subseteq V'$;
- $\gamma_{(S)} \subseteq A'$: the set of arcs $(i, j) \in A'$ such that i and j belong to a set $S \subseteq V'$;
- $\pi(i)$: the vertex in V associated to a vertex $i \in V'$;
- $q(S)$: the sum of the demands of all stations in a set $S \subseteq V_C$, that is, $q(S) = \sum_{i \in \{\pi(j) | j \in S\}} q_i$;
- $f(i)$: the vertex $j \in V'$ that represents the first visit to station $i \in V \setminus \{0\}$. We arbitrarily define $j = \min\{u \in V' \mid \pi(u) = i\}$.

The following decision variables are necessary to define the proposed formulation:

- $x_{ij} = 1$ if arc $(i, j) \in A'$ is in the solution, and 0 otherwise;
- y_{ij}: the flow of bikes on arc $(i, j) \in A'$;
- l_{ij}: the route duration after traversing arc $(i, j) \in A'$;
- s_i: the pickup or delivery performed when visiting node $i \in V'$.

The formulation can be written as follows:

$$\text{minimize} \sum_{a \in A'} c_a x_a \tag{1}$$

subject to

$$\sum_{a \in \delta^+_{(i)}} x_a - \sum_{a \in \delta^-_{(i)}} x_a = 0 \qquad i \in V_C \tag{2}$$

$$\sum_{a \in \delta^+_{(0)}} x_a \leq K \tag{3}$$

$$\sum_{a \in \delta^+_{(0)}} x_a = \sum_{a \in \delta^-_{(d)}} x_a \tag{4}$$

$$\sum_{a \in \delta^+_{(i)}} y_a - \sum_{a \in \delta^-_{(i)}} y_a = s_i \qquad i \in V' \tag{5}$$

$$\sum_{j \in V' : \pi(j) = i} s_j = q_i \qquad i \in V \setminus \{0\} \tag{6}$$

$$\sum_{a \in \delta^+_{(i)}} l_a - \sum_{a \in \delta^-_{(i)}} l_a = \sum_{a \in \delta^+_{(i)}} c_a x_a + s_i h \qquad i \in V_C, q_{\pi(i)} > 0 \tag{7}$$

$$\sum_{a \in \delta^+_{(i)}} l_a - \sum_{a \in \delta^-_{(i)}} l_a = \sum_{a \in \delta^+_{(i)}} c_a x_a - s_i h \qquad i \in V_C, q_{\pi(i)} < 0 \tag{8}$$

$$l_a = c_a x_a + h y_a \qquad a \in \delta^+_{(0)} \tag{9}$$

$$l_a + h y_a \leq L x_a \qquad a \in \delta^-_{(d)} \tag{10}$$

$$\sum_{a \in \delta^-_{(i)}} x_a \leq 1 \qquad i \in V_C \tag{11}$$

$$\sum_{a \in \delta^-_{(f(i))}} x_a = 1 \qquad i \in V \setminus \{0\} \tag{12}$$

$$\sum_{a \in \gamma_{(S)}} x_a \leq |S| - 1 \qquad S \subset V' \tag{13}$$

$$l_a \leq L x_a \qquad a \in A' \tag{14}$$

$$y_a \leq Q x_a \qquad a \in A' \tag{15}$$

$$s_0 = \max\{q_0, 0\} \tag{16}$$

$$s_d = \min\{q_0, 0\} \tag{17}$$

$$s_i \geq \sum_{a \in \delta^+_{(i)}} x_a \qquad i \in V_C, q_{\pi(i)} > 0 \tag{18}$$

$$s_i \leq -\sum_{a \in \delta^+_{(i)}} x_a \qquad i \in V_C, q_{\pi(i)} < 0 \tag{19}$$

$$l_a \geq 0 \qquad a \in A' \tag{20}$$

$$s_i \in \mathbb{Z} \qquad i \in V_C \tag{21}$$

$$y_a \in \mathbb{Z}_{\geq 0} \qquad a \in A'. \tag{22}$$

$$x_a \in \{0, 1\} \qquad a \in A' \tag{23}$$

The objective function (1) minimizes the total travel time. Constraints (2) ensure that if there is an arc arriving at a vertex $i \in V_C$ then there must be an arc leaving the same vertex i. Constraint (3) guarantees that at most K vehicles leave the depot. Constraint (4) enforces the number of vehicles leaving vertex 0 to be the same as that arriving at vertex d. Constraints (5) ensure the flow conservation of bikes for all vertices of the network. Constraints (6) state that the sum of the deliveries or pickups performed at vertices of the network that represent a visit to a given station i should be equal to the demand of i. Constraints (7) and (8) compute the cumulative travel time after visiting station $i \in V_C$, including the handling time. Constraints (9) and (10) determine the departure and arrival time of a route, respectively, which depend on the number of bikes leaving vertex 0 and arriving at vertex d. Constraints (11) forbid a vertex that represents a visit to a station $i \in V_C$ to be visited more than once. Constraints (12) impose a visit to the vertex that represents the first visit to a station $i \in V \setminus \{0\}$. This vertex can be arbitrarily defined; in our case we selected the vertex with the smallest index among those that represent visits to i. Constraints (13) are subtour inequalities. Constraints (14) limit the duration of a route. Constraints (15) prevent the capacity of the vehicle to be exceeded. Constraints (16) and (17) determine the delivery and pickup values of vertices 0 and d, respectively. Constraints (18) and (19) state that there should be at least a delivery or a pickup on every visited performed. Constraints (20)–(22) define the domains of the variables.

Formulation (1)–(22) is not complete since it does not prevent a station to be visited by different vehicles. Let $\mathcal{R}$ be the set composed of all pair of routes, represented by their arc

sets, with at least one station appearing in both of the them. We thus introduce inequalities (24) to forbid the same station from being visited by distinct vehicles.

$$\sum_{a \in \hat{A}} x_a \leq |\hat{A}| - 1 \qquad \hat{A} \in \mathcal{R}. \tag{24}$$

Finally, the above formulation can be strengthened by adding the capacity cuts (25), also utilized by Chemla et al. (2013b):

$$\sum_{a \in \delta^+_{(S)}} x_a \geq \left\lceil \frac{|q(S)|}{Q} \right\rceil \qquad S \subseteq V_C, \; \{\pi(j)|j \in S\} \cap \{\pi(j)|j \notin S\} = \emptyset. \tag{25}$$

Note that condition $\{\pi(j)|j \in S\} \cap \{\pi(j)|j \notin S\}$ in (25) is necessary because it is not valid to consider the demand of a station $i \in V \setminus \{0\}$ if i can be visited outside set S.

3. Branch-and-cut algorithm

Since the mathematical formulation described in Section 2 relies on an exponential number of constraints, we have implemented a branch-and-cut (BC) algorithm. The algorithm initially considers only the polynomial number of constraints (2)–(12) and (14)–(22), and generates the exponential constraints (13), (24) and (25) in a dynamic fashion.

The subtour inequalities (13) are separated using a straightforward min-cut based procedure. The BC algorithm tries to separate them only for the nodes whose depth does not exceed 10, or whenever an integer solution is found. Inequalities (24) can be seen as "no-good" cuts, since they are generally useful to ensure feasibility but they do not strengthen the linear relaxation of the formulation. Therefore, such inequalities are only included when an integer infeasible solution is found. Finally, the capacity cuts (25) are heuristically separated only at the root node by means of the tabu search-based procedure suggested by Augerat et al. (1998), and also implemented by Chemla et al. (2013b).

4. Iterated local search metaheuristic

This section describes the metaheuristic we have developed for the SBRP-MVV. The method is mostly based on iterated local search (ILS) (Lourenço et al., 2010), which alternates between local search (intensification) and perturbation (diversification) moves. ILS has been successfully applied to other vehicle routing problems, especially when

implemented under a multi-start scheme (Subramanian et al., 2010; Penna et al., 2013; Vidal et al., 2015; Silva et al., 2015).

Figure 2 provides an outline of the algorithm. The heuristic performs multiple restarts, where at each of them an ILS-based procedure that executes local search and perturbation moves is iteratively called until n_{ILS} consecutive iterations without improvement have been executed. The best solution of the current multi-start iteration is always the one chosen to be perturbed. Initial solutions are generated using an insertion-based constructive heuristic (see Section 4.3) that allows infeasible solutions. However, if a feasible solution is not found after $2n_R$ trials, then the algorithm generates a new initial solution. The algorithm terminates after n_R feasible restarts or after $2n_R$ restarts. The local search is executed using a randomized variable neighborhood decent (RVND) procedure (see Section 4.4), whereas the perturbation procedure performs random shift, swap or split moves (see Section 4.5).

4.1. Auxiliary data structures

We have implemented some auxiliary data structures (ADSs), following the ideas presented in Hernández-Pérez and Salazar-González (2004), in order to improve the performance of the proposed heuristic both in terms of computational complexity and of the total number of operations performed.

Let $\sigma = (\sigma_{(0)}, \ldots, \sigma_{(|\sigma|-1)})$ be a subsequence of a solution S (with $\overleftarrow{\sigma}$ as the associated reverse subsequence), and let $\sigma_{i,j}$ be the subsequence of σ that starts at the i^{th} position and ends at the j^{th} position, i.e., $\sigma_{i,j} = (\sigma_{(i)}, \ldots, \sigma_{(j)})$. Moreover, let $q'_{\sigma_{(i)}}$ be the load delivered or collected at station $\sigma_{(i)}$ *on that particular visit*. For each possible σ of S the method stores and updates:

- $q_{sum}(\sigma) = \sum_{i=0}^{|\sigma|-1} q'_{\sigma_{(i)}}$ = sum of the loads delivered/collected (cumulative load);
- $q_{min}(\sigma) = \min\{0, q_{sum}(\sigma_{0,0}), q_{sum}(\sigma_{0,1}), \ldots, q_{sum}(\sigma_{0,|\sigma|-1})\}$ = minimum cumulative load;
- $q_{max}(\sigma) = \max\{0, q_{sum}(\sigma_{0,0}), q_{sum}(\sigma_{0,1}), \ldots, q_{sum}(\sigma_{0,|\sigma|-1})\}$ = maximum cumulative load;
- $l_{min}(\sigma) = -q_{min}(\sigma)$ = minimum flow of bikes allowed to enter σ so as to ensure feasibility or so that the infeasibility does not increase;
- $l_{max}(\sigma) = Q - q_{max}(\sigma)$ = maximum flow of bikes allowed to enter σ so as to ensure feasibility or so that the infeasibility does not increase;

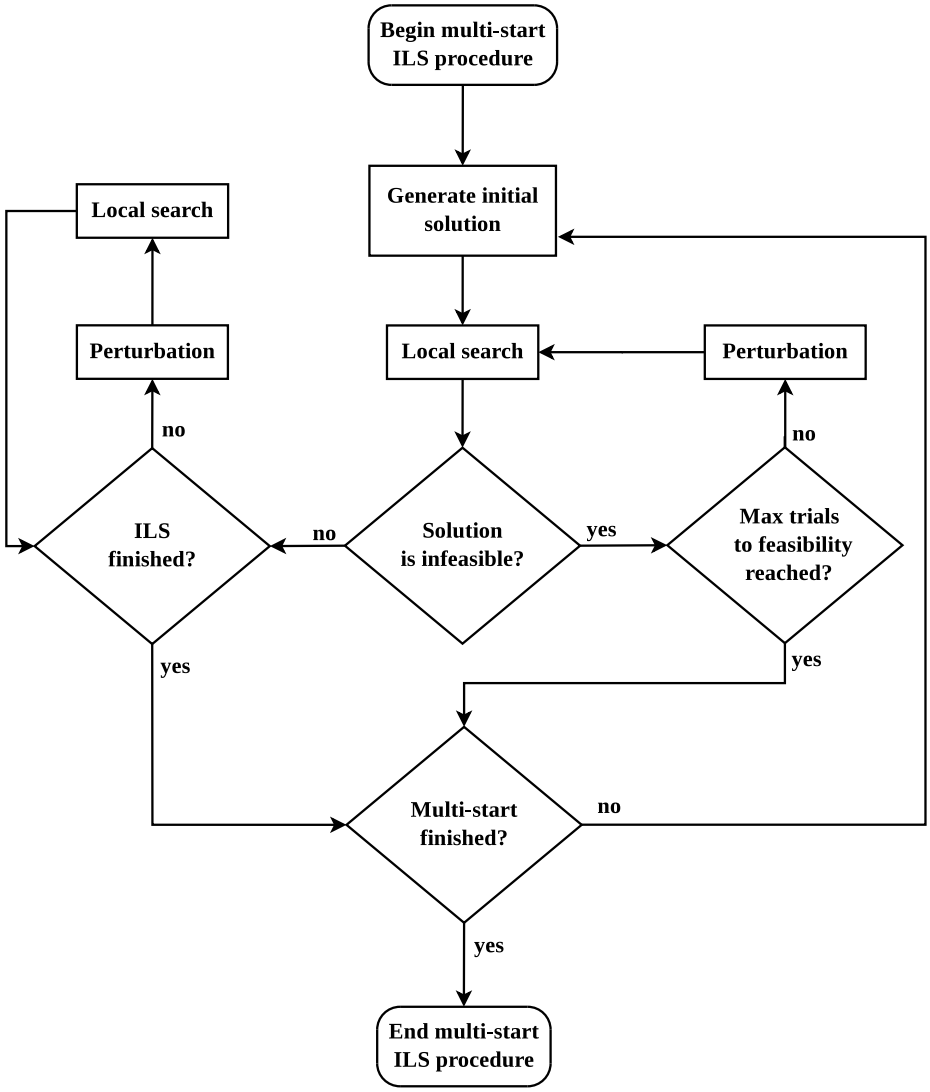


Figure 2: Multi-start ILS flowchart

- $tt(\sigma) = \sum_{i=0}^{|\sigma|-2} \sum_{j=i+1}^{|\sigma|-1} c_{\sigma_{(i)}\sigma_{(j)}}$ = travel time;
- $dur(\sigma) = tt(\sigma) + h \sum_{i=0}^{|\sigma|-1} |q'_{\sigma_{(i)}}|$ = total duration (travel time + handling time).

Note that it is not necessary to store the ADSs regarding $\overleftarrow{\sigma}$ because they can be directly derived in constant time from those stored for σ:

- $q_{sum}(\overleftarrow{\sigma}) = q_{sum}(\sigma)$;
- $q_{min}(\overleftarrow{\sigma}) = q_{sum}(\sigma) - q_{max}(\sigma)$;
- $q_{max}(\overleftarrow{\sigma}) = q_{sum}(\sigma) - q_{min}(\sigma)$;

- $l_{min}(\overleftarrow{\sigma}) = -q_{sum}(\sigma) + q_{max}(\sigma)$;
- $l_{max}(\overleftarrow{\sigma}) = Q - q_{sum}(\sigma) + q_{min}(\sigma)$;
- $tt(\overleftarrow{\sigma}) = tt(\sigma)$ (assuming that $c_{ij} = c_{ji}$);
- $dur(\overleftarrow{\sigma}) = dur(\sigma)$ (assuming that $c_{ij} = c_{ji}$).

When a subsequence σ' is composed of only one station (or depot) v, then $q_{sum}(\sigma') = q'_v$, $q_{min}(\sigma') = \min(0, q'_v)$,$q_{max}(\sigma') = \max(0, q'_v)$, $l_{min}(\sigma') = -q_{min}(\sigma')$, $l_{max}(\sigma') = Q - q_{max}(\sigma')$, $tt(\sigma') = 0$ and $dur(\sigma') = h|q'_v|$. Let the operator $\oplus$ denote the concatenation of two distinct subsequences. In what follows we show that any subsequence σ, $|\sigma| > 1$, can be derived from two other subsequences σ^1 and σ^2 by means of the concatenation operator $\oplus$:

$$q_{sum}(\sigma^1 \oplus \sigma^2) = q_{sum}(\sigma^1) + q_{sum}(\sigma^2); \tag{26}$$
$$q_{min}(\sigma^1 \oplus \sigma^2) = \min\{q_{min}(\sigma^1), q_{sum}(\sigma^1) + q_{min}(\sigma^2)\}; \tag{27}$$
$$q_{max}(\sigma^1 \oplus \sigma^2) = \max\{q_{max}(\sigma^1), q_{sum}(\sigma^1) + q_{max}(\sigma^2)\} \tag{28}$$
$$l_{min}(\sigma^1 \oplus \sigma^2) = -q_{min}(\sigma^1 \oplus \sigma^2); \tag{29}$$
$$l_{max}(\sigma^1 \oplus \sigma^2) = Q - q_{max}(\sigma^1 \oplus \sigma^2); \tag{30}$$
$$tt(\sigma^1 \oplus \sigma^2) = tt(\sigma^1) + c_{\sigma^1_{(|\sigma^1|-1)}\sigma^2_{(0)}} + tt(\sigma^2); \tag{31}$$
$$dur(\sigma^1 \oplus \sigma^2) = dur(\sigma^1) + c_{\sigma^1_{(|\sigma^1|-1)}\sigma^2_{(0)}} + dur(\sigma^2). \tag{32}$$

The total number of subsequences of a solution S is of the order of $|V|^2$. Since the information stored for each subsequence can be updated in constant time, it takes $\mathcal{O}(|V|^2)$ operations to update all ADSs.

We now provide an example (see Figure 3). Let $\sigma = (2, 1, 3, 4, 1)$ be a subsequence involving four stations (1, 2, 3 and 4) and five visits with $q'_{\sigma_{(0)}} = 3$, $q'_{\sigma_{(1)}} = -3$, $q'_{\sigma_{(2)}} = 4$, $q'_{\sigma_{(3)}} = -2$, and $q'_{\sigma_{(4)}} = -1$. The vehicle collects three bikes at station 2, delivers three bikes at station 1, collects four bikes at station 3, delivers two bikes at station 4, and finally delivers one bike at station 1. Note that station 1 is visited twice. Moreover, assume that $Q = 5$, $h = 2$, $c_{21} = 2$, $c_{13} = 2$, $c_{34} = 1$ and $c_{41} = 2$. We therefore have the following values for the ADSs for this subsequence:

- $q_{sum}(\sigma) = 3 - 3 + 4 - 2 - 1 = 1$;
- $q_{min}(\sigma) = \min(0, 3, 0, 4, 2, 1) = 0$;
- $q_{max}(\sigma) = \max(0, 3, 0, 4, 2, 1) = 4$;

- $l_{min}(\sigma) = 0$;
- $l_{max}(\sigma) = 5 - 4 = 1$;
- $tt(\sigma) = 2 + 2 + 1 + 2 = 7$;
- $dur(\sigma) = 7 + 2(3 + 3 + 4 + 2 + 1) = 33$.

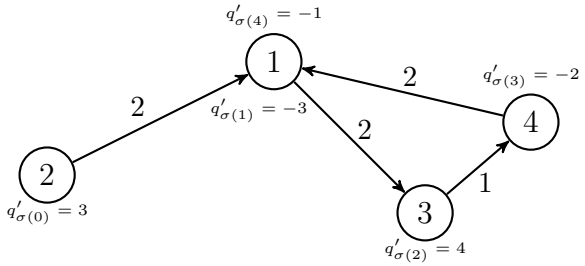


Figure 3: Example of a subsequence involving 4 stations and 5 visits.

Note that the number of bikes that can enter σ, so as to ensure feasibility, must lie within the interval $[l_{min}(\sigma), l_{max}(\sigma)] = [0, 1]$. Suppose now that $Q = 3$. This leads to an infeasibility ($[l_{min}(\sigma), l_{max}(\sigma)] = [0, -1]$) since the capacity of the vehicle would be exceeded when visiting station 3.

4.2. Evaluation function

Let w_Q, w_L and w_N be the penalty weights associated with violations on load, route duration and number of visits, respectively. We also define $V(\sigma)$ as the set of stations that are part of a route σ, and n_i as the number of visits to a station $i \in V(\sigma)$. The evaluation function of route σ, which can be seen as a subsequence starting and ending at the depot, is given by

$$Z(\sigma) = tt(\sigma) + w_Q(\max\{0, -q_{min}(\sigma)\} + \max\{0, q_{max}(\sigma) - Q\}) + w_L(\max\{0, dur(\sigma) - L\}) + w_N \sum_{i \in V(\sigma)} (\max\{0, n_i - N\}). \tag{33}$$

The first term of the right-hand side of Equation (33) measures the travel time of the route. The second one computes the penalty regarding the maximum violation on the vehicle load. The violation occurs when the load exceeds the vehicle capacity or when the load becomes negative. The third term is the penalty incurred when the route duration

exceeds the maximum limit. Finally, the last term computes the penalty in case of violation on the maximum number of visits.

The values of w_Q, w_L and w_N are initially set to 1000, 10 and 100, respectively. If the local search returns an infeasible solution, then the penalty coefficients are automatically adjusted as follows:

- in case of an infeasibility due to load: $w_Q = \min\{100000, 1.2w_Q\}$; otherwise, $w_Q = \max\{1000, 0.8w_Q\}$;
- in case of an infeasibility due to route duration: $w_L = \min\{1000, 1.2w_Q\}$; otherwise, $w_L = \max\{10, 0.8w_Q\}$;
- in case of an infeasibility due to visits, $w_N = \min\{10000, 1.2w_Q\}$; otherwise, $w_N = \max\{100, 0.8w_Q\}$.

4.3. Constructive procedure

Algorithm 1 describes the insertion procedure used to build an initial solution. Let R be the set composed of K initially empty routes and let SL be a set composed of the stations whose demands have not been fully met in the partial solution. Firstly, all stations are considered to be part of SL (lines 7–8). We define S as the partial initial solution and CL as the candidate list composed of tuples containing information regarding each insertion. A tuple has the format $(v, p, r, \Delta, q', \delta)$, where v is the station, p is the position in which v would be inserted in a given route r, Δ is the insertion cost of v in the associated position p and route r, q' is the load to be delivered ($q' < 0$) or collected ($q' > 0$) in v, and δ is the variation on the number of bikes at the beginning of the route. In addition, let $rd(v)$ be the residual demand of station v and let $rd(0)$ be the residual demand of the depot.

While SL is not empty, the candidate list of all possible insertions is built at every iteration (lines 10–30). The load q' to be delivered or collected during a particular visit is determined by an auxiliary procedure called `DecideStationLoad`. This procedure not only specifies the value of q', but also the value of δ (line 14). In a first round, the algorithm only considers feasible insertions (lines 15–16). However, when this is no longer possible (lines 20–22), the total residual demand of a station is met in a single yet infeasible visit (lines 17–19). Next, the insertion cost is computed and CL is updated (lines 18–19). Note that a station is only allowed to be inserted in route r if it has not yet been added to the partial solution S or if it has been already added to route r itself. In practice, this is to prevent the same station from being visited by distinct routes.

Algorithm 1 Constructive procedure

1: **procedure** BuildInitialSolution(seed, data)
2: $SL \leftarrow V \setminus \{0\}$
3: $S \leftarrow \emptyset$
4: $firstRound \leftarrow true$
5: $l_{depot} \leftarrow \max(0, rd(0))$ /*Load of the depot*/
6: $l_{ini}(r) \leftarrow 0, \ \forall r \in R$ /*Initial load of route r*/
7: **for** each station $v \in V \setminus \{0\}$ **do**
8: $SL \leftarrow v$
9: **while** $|SL| > 0$ **do**
10: $CL \leftarrow \emptyset$
11: **for** each $v \in SL$ **do**
12: **for** each $r \in R$ in which v can be inserted **do**
13: **for** each position p of route r **do**
14: $[q', \delta] \leftarrow$ DecideStationLoad$(v, p, r, rd(v), l_{ini}(r), l_{depot})$ /*See Alg. 2*/
15: **if** $q' = 0$ **and** $firstRound = false$ **then**
16: $q' \leftarrow rd(v)$
17: **if** $q' \neq 0$ **then**
18: $\Delta \leftarrow$ cost of inserting v in position p of r
19: $CL \leftarrow (v, p, r, \Delta, q', \delta)$
20: **if** $|CL| = 0$ **then**
21: $firstRound \leftarrow false$
22: Go to line 9
23: $g \leftarrow$ element from CL selected using the idea of the constructive phase of GRASP
24: $S \leftarrow S \cup g(v)$ in position $g(p)$ of route $g(r)$ with load $g(q')$
25: $l_{ini}(g(r)) \leftarrow l_{ini}(g(r)) + g(\delta)$
26: $l_{depot} \leftarrow l_{depot} - g(\delta)$
27: $rd(v) \leftarrow rd(v) - g(q')$
28: **if** $rd(v) = 0$ **then**
29: $SL \leftarrow SL \setminus \{v\}$
30: Update ADSs
31: Check for consecutive visits to the same station in a route and, if it is the case, merge them so that only a single visit is performed
32: **return** S
33: **end** BuildInitialSolution

Once CL is built, one element is selected using the same idea of the construction phase of the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic (Resende and Ribeiro, 2010) (line 23). More precisely, a restricted candidate list (RCL) is created by choosing the elements from CL associated with the best insertions. The size of RCL is controlled by a parameter α that defines the level of greediness or randomness.

The larger the value of α, the larger the size of RCL. In our experiments α is selected at random from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. An element is then randomly chosen from RCL and the associated station v is inserted into the partial solution S on the specified position p of route r with the corresponding load q' (line 24). The initial load of route $g(r)$ as well as the load of the depot are updated in lines 25–26. Next, the residual demand of station v is updated and, in case it turns out to be zero, v is removed from SL (lines 27–29). Moreover, all ADSs are updated accordingly (line 30).

After all vertices have been inserted and their respective demands are fully met, the algorithm checks whether a pair of identical vertices appear consecutive to each other in a route. If so, these are merged into a single visit (line 31). Finally, the initial solution is returned (line 32).

Algorithm 2 presents the auxiliary procedure employed to determine the number of bikes to be delivered or collected at station v when considering an insertion in a possible position p of route r. The decision is performed based (i) on the cumulative load and load intervals of the subsequence of r starting from the first visit and ending at position $p-1$, here denoted as subsequence σ^1; (ii) on the load intervals of the subsequence of r starting from the station associated with position p and ending at the last visit, here denoted as subsequence σ^2; (iii) on the extra load initially available at the depot (l^+) that may be used to meet the residual demand of a delivery station v without violating the loading constraints of σ^1; (iv) on the extra number of bikes that may be brought back to the depot (l^-) due to the insertion of a pickup station v. Note that these informations can be accessed in constant time by simply checking the values of $q_{sum}(\sigma^1)$, $l_{min}(\sigma^1)$, $l_{min}(\sigma^2)$ and $l_{max}(\sigma^2)$.

If a vehicle leaving σ^1 enters σ^2 with more bikes than the minimum limit required by σ^2, i.e, $q_{sum}(\sigma^1) + l_{ini}(r) + l^+ > l_{min}(\sigma^2)$, it is then possible to insert a delivery station v ($rd(v) < 0$) in position p with a corresponding load q' given by the maximum between the excess of bikes of the vehicle and $rd(v)$ (lines 5–10). Similarly, if a vehicle leaving σ^1 enters σ^2 with fewer bikes than the maximum limit required by σ^2, i.e, $q_{sum}(\sigma^1) + l_{ini}(r) - l^- < l_{max}(\sigma^2)$, it is then possible to insert a pickup station v ($rd(v) > 0$) in position p with a corresponding load q' given by the minimum between the residual capacity of the vehicle and $rd(v)$ (lines 11–16). The variation on the depot load is computed in lines 9–10 and 15–16.

4.4. Local search

RVND (Subramanian et al., 2010; Subramanian, 2012) extends the well-known VND procedure (Mladenović and Hansen, 1997) by allowing a random ordering of the neighbor-

Algorithm 2 Decide the load of a station to be inserted in position p of a route r

procedure DecideStationLoad($v, p, r, rd(v), l_{ini}(r), l_{depot}$)
Let σ^1 be the subsequence of r starting from the first visit of a the route r and ending at the station associated with position $p-1$
Let σ^2 be the subsequence of r starting from the station associated with position p and ending at the last visit of route r
$q' \leftarrow 0$; $\delta \leftarrow 0$
if $rd(v) < 0$ **then**
    $l^+ \leftarrow max(0, min(l_{depot}, l_{max}(\sigma^1) - l_{ini}(r)))$ */*Extra load available at the depot that does not violate σ^1*/*
    if $q_{sum}(\sigma^1) + l_{ini}(r) + l^+ > l_{min}(\sigma^2)$ **then**
        $q' \leftarrow -min(q_{sum}(\sigma^1) + l_{ini}(r) + l^+ - l_{min}(\sigma^2), |rd(v)|)$
        if $|q'| > max(0, q_{sum}(\sigma^1) + l_{ini}(r) - l_{min}(\sigma^2))$ **then**
            $\delta \leftarrow |q'| - max(0, q_{sum}(\sigma^1) + l_{ini}(r) - l_{min}(\sigma^2))$
else
    $l^- \leftarrow max(0, l_{ini}(r) - l_{min}(\sigma^1))$ */*Extra load returning to the depot*/*
    if $q_{sum}(\sigma^1) + l_{ini}(r) - l^- < l_{max}(\sigma^2)$ **then**
        $q' \leftarrow min(l_{max}(\sigma^2) - (q_{sum}(\sigma^1) + l_{ini}(r) - l^-), rd(v))$
        if $q' > max(0, l_{max}(\sigma^2) - (q_{sum}(\sigma^1) + l_{ini}(r)))$ **then**
            $\delta \leftarrow max(0, l_{max}(\sigma^2) - (q_{sum}(\sigma^1) + l_{ini}(r))) - q'$
return $[q', \delta]$
end DecideStationLoad

hoods. All possible moves of each neighborhood are examined and the search resumes from the best improving neighbor. If a neighborhood is not capable of finding an improved solution, the procedure then selects another one at random.

We consider two classes of neighborhood structures: inter-route and intra-route. The first performs moves between a pair of routes, whereas the latter only consider moves within the same route. The main RVND scheme considers only inter-route neighborhoods. Intra-route neighborhoods are only applied, also in an RVND fashion, over those routes that have been affected by an inter-route move or by a perturbation move.

As in the construction phase, if the local search procedure detects that there are two or more consecutive visits to the same station in a modified solution, they are then merged into a single visit. It should be also pointed out that the ADSs are only updated for the subsequences that were modified by a move. Finally, all moves are evaluated in amortized constant time by using the information stored in the ADSs.

4.4.1. Inter-route neighborhood structures

The following inter-route neighborhood structures were implemented:

- Shift$(\sigma, 0)$: a subsequence σ is moved from one route to another one. We have limited

the size of σ to 1 and 2. Each size is assumed to be a different neighborhood in the RVND. Only those subsequences that do not have stations visited multiple times in the route are considered.

- Swap(σ^1, σ^2): subsequence σ^1 from one route is interchanged with another subsequence σ^2 from a different route. We have limited the size of σ^1 and σ^2 to 1 and 2. Disregarding symmetries, each of the three possibilities is assumed to be a different neighborhood in the RVND. As in the previous case, only those subsequences containing stations with a single visit in the respective route are considered.

- 2-opt*: two distinct routes are divided into two subsequences each: σ^1 and σ^2, for the first route, and σ^3 and σ^4, for the second one. Next, two new routes are derived by connecting σ^1 with σ^4 and σ^3 with σ^2.

4.4.2. Intra-route neighborhood structures

The following intra-route neighborhood structures were implemented:

- Reinsertion(σ): a subsequence σ is removed and reinserted in another position of the route. The size of σ was limited to 1, 2 and 3, thus resulting in three different neighborhoods.

- Exchange(σ^1, σ^2): subsequence σ^1 is interchanged with other subsequence σ^2. The size of σ^1 and σ^2 were restricted to 1 and 2, thus leading to three distinct neighborhoods (disregarding symmetries).

- 2-opt: an arc is removed and another one is inserted so as to build a new route.

- Split: a visit is selected and then a copy of the station associated with such visit is inserted in another position (non-adjacent to the original visit) of the route. All split possibilities are considered.

4.5. Perturbation mechanisms

The perturbation procedure randomly selects one of the following mechanisms:

- Multiple shift$(\sigma, 0)$: multiple Shift$(\sigma, 0)$ moves are applied at random and we limited $|\sigma|$ to be at most 3.

- Multiple swap(σ^1, σ^2): multiple Swap(σ^1, σ^2) moves are applied at random. In our case we limited $|\sigma^1| = |\sigma^2|$ to be at most 3.

- Split in half: this is a particular case of the Split neighborhood, where the value of the original load is equally split between the original visit and the copy, and the latter is inserted in a random position (non-adjacent to the original visit) of the route. In this case the mechanism selects the visit associated with the largest delivery/pickup of a random route.

5. Computational experiments

The BC algorithm was implemented using CPLEX 12.6 over the mathematical formulation described in Section 2. This algorithm and the ILS based heuristic presented in Section 4 were both coded in C++ and they were executed on a Intel Xeon E5-2650 v2 processor with a clock speed of 2.60 GHz and 64 GB of RAM, running on Scientific Linux release 6.5 (Carbon). We have considered only a single thread when running the algorithms. ILS was run 10 times for each instance. The best solution found by ILS was provided for the BC as an initial primal bound.

5.1. Instances

The SBRP-MVV instances were derived from those proposed by Hernández-Pérez and Salazar-González (2004) for the one-commodity pickup and delivery traveling salesman problem (1-PDTSP), which are available at `http://hhperez.webs.ull.es/PDsite/#Benchmark`. We have considered a total of 630 instances involving up to 200 vertices and, for each of them, we have adopted two distinct values for the number of vehicles and for the maximum number of visits allowed, namely $K = \{2, 3\}$ and $N = \{2, 3\}$. The route duration limit for each instance was computed as follows: $L = \lceil f \times \overline{LB}/K \rceil$, where $\overline{LB}$ is computed by solving the LP relaxation of the model SBRP-R of Erdoğan et al. (2015) and f is the minimum value in the set $\{1, 1.1, 1.2, 1.3, \dots\}$ for which an "aggressive" version of the heuristic is capable of finding a feasible solution. Moreover, we have adopted $h = 10$ for all instances.

Preliminary experiments revealed that a station is seldom visited more than twice, even for those instances with $Q = 10$. Therefore, we decided to disregard the instances with $N = 3$, leading to a total of $2 \times 630 = 1260$ instances.

We have also tested our algorithms on the real-world instances of Rainer-Harbach et al. (2015) (available at `https://www.ac.tuwien.ac.at/research/problem-instances/#bbss`). In this case, the value of the duration limit is the same for all instances, i.e., $L = 480$, and we have imposed $N = 2$. For each instance a meaningful number of vehicles was computed so as to ensure feasibility. The original instances ranged from 10 to 700

vertices. Nevertheless, because the BC procedure was only worth to be ran on instances with up to 180 vertices, we chose to run the proposed algorithms on the groups of instances involving 10, 20, 30, 60, 90, 120 and 180 vertices. Since each group has 30 test-problems, this leads to a total of $7 \times 30 = 210$ instances.

Finally, we have adapted the 65 real-world instances proposed in Dell'Amico et al. (2014) (available at http://www.or.unimore.it/resources/BRP/home.html) to our problem by also adopting $h = 10$ and imposing a route duration limit as described above. In this case, the value of K for each instance of this benchmark dataset was defined as the number of vehicles of the best-known solution. Furthermore, as opposed to the 1-PDTSP instances, the test problems suggested are unbalanced, that is, the sum of the demands need not be zero. Therefore, in order to balance the instances, we have set $q_0 = -\sum_{j \in V \setminus \{0\}} q_j$. The size of the instances ranges from 13 to 116 vertices.

5.2. Parameter tuning

Regarding the parameter n_{ILS}, we set its value as a function of the instance, more precisely, $n_{ILS} = \max(I_{min}, |V|)$, where I_{min} is an input parameter used to avoid an insufficient number of perturbations for small size instances. We then followed the procedure described by Cruz et al. (2017) to calibrate the parameter I_{min}. For this testing we arbitrarily set $n_R = 1$ and we adopted $|\sigma| = 1$ for neighborhoods Shift(σ,0) and Reinsertion(σ), $|\sigma^1| = |\sigma^2| = 1$ for neighborhood Swap(σ^1, σ^2), and $|\sigma^1| = |\sigma^2| \leq 2$ for neighborhood Exchange(σ^1, σ^2). After performing some experiments on a subset of 30 random instances, containing two instances with capacity values in $\{10, 15, 20\}$ for each size in $\{20, 40, 60, 100, 200\}$, we set $I_{min} = 100$. We chose instances with tight capacity values because they are likely to be more harder to solve.

Once we set $n_{ILS} = \max\{100, |V|\}$, we performed a series of experiments to calibrate the parameter n_R, and to determine the neighborhood structures to be used in the local search. We tested three different values for n_R and for each of them we tried four different types of neighborhood combinations. Note that because there are 30 instances and two values for K, then each scenario contains $30 \times 2 = 60$ instances. Since we executed the algorithm 10 times for each setting, the total number of runs for each setting is equal to 600. The percentage of feasible solutions returned by the algorithm in the 600 runs for each setting, as well as the average gap between the average solutions and the best solution found during the tuning phase are reported in Table 2. The latter was computed considering only those instances for which all settings could find feasible solutions in all 10 runs. From the results obtained, we decided to set $n_R = 20$ and the neighborhoods selected were Shift(1,1),

Swap(1,1), 2-opt*, Reinsertion(1), Exchange(1,1), Exchange(2,2), 2-opt and Split.

Table 2: Percentage of feasible runs and average solution cost (out of 600) for each setting

Local Search Configuration	$n_R = 10$		$n_R = 15$		$n_R = 20$	
	Feas (%)	Avg Gap (%)	Feas (%)	Avg Gap (%)	Feas (%)	Avg Gap (%)
Shift(1,1) + Swap(1,1) + 2-opt* Reinsertion(1) + Exchange(1,1) + 2-opt + Split	71.17	0.82	83.00	0.51	90.33	0.33
Shift(1,1) + Swap(1,1) + 2-opt* Reinsertion(1) + Exchange(1,1) + 2-opt + Split + Reinsertion(2,2)	72.00	0.69	86.50	0.50	92.83	0.26
Shift(1,1) + Swap(1,1) + 2-opt* Reinsertion(1) + Exchange(1,1) + 2-opt + Split + Exchange(2,2)	75.00	0.78	85.33	0.43	92.00	0.31
Shift(1,1) + Swap(1,1) + 2-opt* + Reinsertion(1) + Exchange(1,1) + 2-opt + Reinsertion(2,2) + Exchange(2,2) + Split	73.83	0.72	89.33	0.36	**92.67**	**0.18**

5.3. Evaluating the impact of Q, N and K on solving the SBRP-MVV

In this section we are interested in evaluating the impact of the vehicle capacity and of the number of vehicles on the number of visits to a station, as well as on the average gap, defined as the ratio of the difference of the best upper and lower bounds divided by the best lower bound. We also examine the benefits and disadvantages of allowing multiple visits, in terms of solution quality and CPU time, according to the vehicle capacity.

Figure 4 shows the percentage of instances for which the best solution found by ILS required multiple visits to the same station according to the value of Q. We can verify that the percentage of instances requiring multiple visits tends to increase as the capacity of the vehicle decreases. This observation becomes more evident for $Q = \{10, 15, 20\}$. Moreover, it seems that multiple visits are more frequent for $K = 2$ than for $K = 3$.

Figure 5 illustrates the average gap between the best solutions found by ILS and the lower bounds obtained by BC for the different values of Q. It suggests that the instances appear to become more challenging when there are more vehicles available. However, it is important to point out that the value of L decreases as the number of vehicle increases (see Section 5.1), which may potentially contribute to increase the level of difficulty of solving the SBRP-MVV.

From the results presented in Figures 4 and 5 it is possible to verify that multiple visits seem to be less attractive for $Q \geq 25$. We thus hereafter only report the results for the instances with $Q \leq 20$.

Figure 6 depicts the average percentage improvement on the best solution obtained by allowing multiple visits. Our aim in this case is to estimate the benefits of letting a station

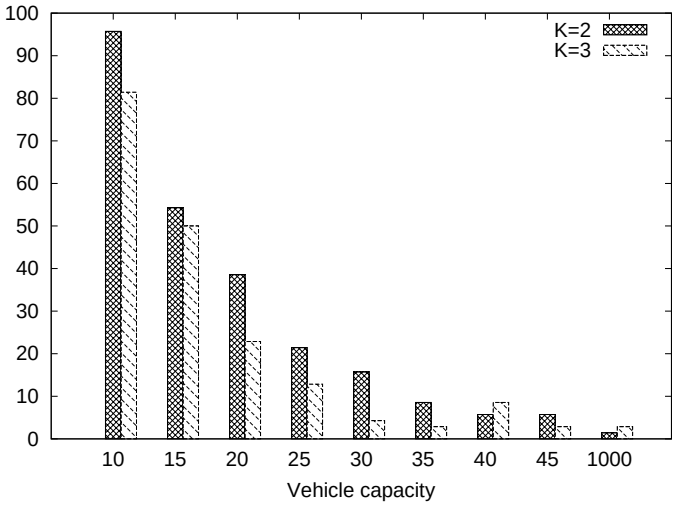


Figure 4: Percentage of instances for which the best solution found required multiple visits to the same station

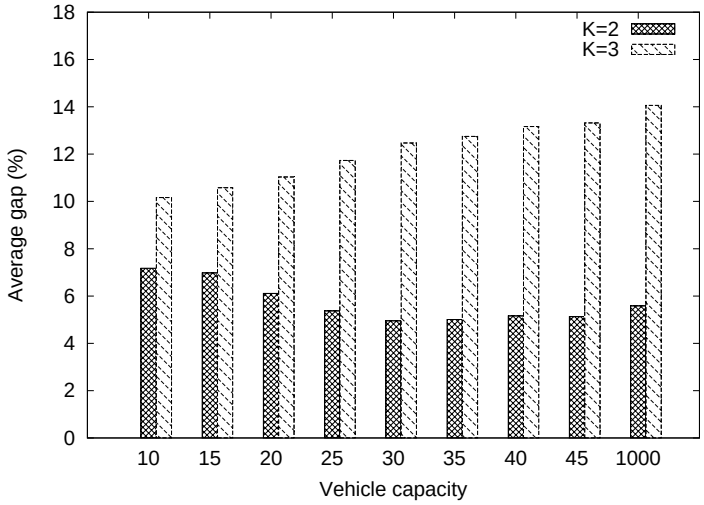


Figure 5: Average gaps between the best solution found by ILS and the lower bound obtained by BC

to be visited more than once on the quality of the best solution found by ILS as the vehicle capacity increases. The results suggest that the improvement is considerable for $Q = 10$ and still significant for $Q = 15$ and $Q = 20$, but only for $K = 2$ in the latter case. In addition, these are consistent with the results shown in Figure 4, that is, the number of vehicles has an impact on the need to make multiple visits for finding high quality solutions.

Finally, Figure 7 shows the impact on CPU time of ILS when allowing multiple visits

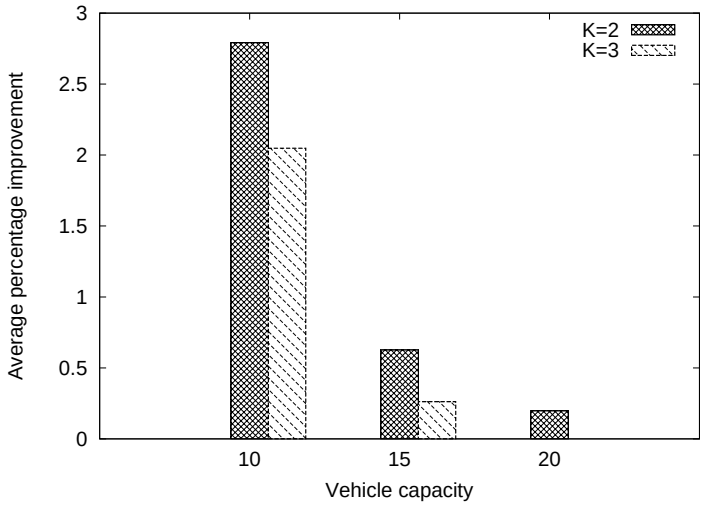
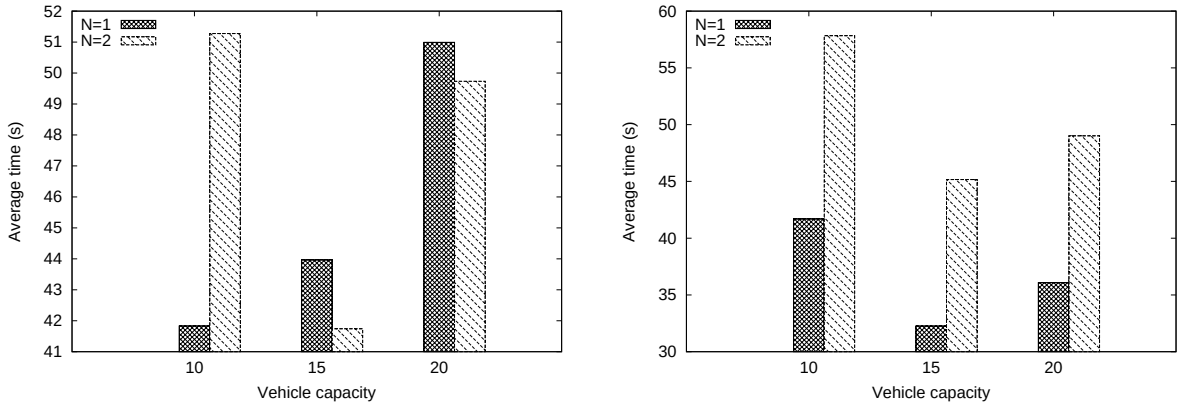


Figure 6: Average percentage improvement on the best solution obtained by allowing multiple visits

to a station. For $K = 3$, there is a clear runtime disadvantage when the algorithm tries to exploit the possibility of visiting stations more than once. In the case of $K = 2$, the increase in the CPU time is more perceptible only for $Q = 10$, whereas for $Q = 15$ and $Q = 20$ the runtime performance is more or less similar.



(a) $K = 2$ (b) $K = 3$

Figure 7: Average CPU time spent by ILS

5.4. Detailed results for instances with 20 and 30 vertices

Tables 3–6 present the detailed results found by the BC and ILS algorithms for the instances involving 20 and 30 vertices. Regarding the BC results, **Root LB** denotes the

lower bound obtained after solving the root node, **LB** corresponds to the best lower bound found, **Root time (s)** corresponds to the CPU time in seconds spent to solve the root node, **Time (s)** is the total CPU time in seconds, **Tree size** indicates the number of nodes of the BC tree, **#Lazy Cuts**, **#Subtour Cuts** and **#Cap. Cuts** report the number of lazy cuts (24), subtour cuts (13) and capacity cuts (25) added, respectively. For what concerns the ILS results, **Min cost** denotes the best cost found in the 10 runs, **Avg Cost** corresponds to the average cost of the 10 runs, **Avg Gap (%)** indicates the average gap between the solution value and the lower bound, and **Avg Time (s)** represents the average CPU time of the 10 runs.

Table 3: Summary of computational results for $|V| = 20$, $K = 2$ and $N = 2$

	BC								ILS			
Instance	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n20q10A	4721.77	**5006.00**	2.57	628.30	88679	0	31	228	**5006**	5006.00	0.00	0.57
n20q10B	4897.08	**5052.00**	1.36	25.37	2046	44	21	97	**5052**	5052.00	0.00	0.68
n20q10C	6060.18	**6118.00**	4.08	30.40	951	0	10	219	**6118**	6118.00	0.00	1.09
n20q10D	6127.35	**6277.00**	4.37	132.41	10460	522	22	255	**6277**	6277.00	0.00	0.83
n20q10E	6320.66	**6381.00**	5.01	46.71	4964	0	7	236	**6381**	6381.00	0.00	0.89
n20q10F	4731.55	**4983.00**	4.01	41.41	2462	0	16	175	**4983**	4983.00	0.00	0.78
n20q10G	5270.01	5485.15	3.77	3600.00	400768	17	34	160	5788	5788.00	5.52	0.57
n20q10H	5586.39	5868.00	3.19	3600.00	228658	62	35	206	6328	6328.00	7.84	0.62
n20q10I	4699.58	**4979.00**	3.90	1314.72	121218	67	25	172	**4979**	4979.00	0.00	0.74
n20q10J	4256.31	4572.31	2.10	3600.00	359551	1562	71	147	4995	4995.00	9.24	0.74
n20q15A	4071.24	**4435.00**	2.50	311.64	39425	2	27	117	**4435**	4435.00	0.00	0.50
n20q15B	4435.77	**4740.00**	1.97	52.73	7856	0	17	95	**4740**	4740.00	0.00	0.50
n20q15C	5240.41	**5494.00**	4.77	209.86	17105	105	7	139	**5494**	5494.00	0.00	0.76
n20q15D	5501.72	**5638.00**	4.02	89.02	9710	0	21	153	**5638**	5638.00	0.00	0.69
n20q15E	5565.47	**5800.00**	2.96	316.83	40177	24	25	237	**5800**	5800.00	0.00	0.70
n20q15F	4572.32	**4923.00**	3.10	247.51	50063	16	22	108	**4923**	4923.00	0.00	0.74
n20q15G	4747.97	**5218.00**	3.20	962.98	192043	0	52	110	**5218**	5218.00	0.00	0.60
n20q15H	4780.69	5377.77	2.91	3600.00	376378	3237	47	167	5635	5635.00	4.78	0.66
n20q15I	4196.48	**4615.00**	2.30	791.95	145927	28	34	86	**4615**	4615.00	0.00	0.74
n20q15J	4017.22	**4270.00**	2.16	33.64	2008	0	12	108	**4270**	4270.00	0.00	0.53
n20q20A	3775.47	4217.22	2.61	3600.00	412869	2	56	51	4435	4435.00	5.16	0.50
n20q20B	4376.33	**4740.00**	1.98	37.79	3398	0	14	61	**4740**	4740.00	0.00	0.53
n20q20C	4631.64	4949.28	3.38	3600.00	301521	0	41	111	5203	5203.00	5.13	0.70
n20q20D	4723.41	**5016.00**	3.66	420.76	61895	266	38	115	**5016**	5016.00	0.00	0.69
n20q20E	4756.93	**4763.00**	1.97	2.05	2	0	3	135	**4763**	4763.00	0.00	0.51
n20q20F	4425.36	**4674.00**	2.28	34.93	1201	0	20	81	**4674**	4674.00	0.00	0.65
n20q20G	4571.55	5050.09	2.68	3600.00	454400	2	69	77	5256	5256.00	4.08	0.61
n20q20H	4371.88	**5320.00**	1.54	867.22	146825	479	40	54	**5320**	5320.00	0.00	0.58
n20q20I	4154.69	**4640.00**	2.11	2525.64	343159	2	52	113	**4640**	4640.00	0.00	0.70
n20q20J	3862.59	**4145.00**	2.30	25.20	1050	0	9	112	**4145**	4145.00	0.00	0.47

Table 4: Summary of computational results for $|V| = 30$, $K = 2$ and $N = 2$

Instance	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n30q10A	6231.44	6348.43	23.99	3600.00	30414	0	46	314	6552	6552.00	3.21	1.67
n30q10B	6418.54	6499.57	19.29	3600.00	129093	762	20	351	6617	6617.00	1.81	1.63
n30q10C	6202.49	6400.12	20.14	3600.00	24143	0	37	289	6625	6625.40	3.52	2.32
n30q10D	6009.62	**6219.00**	24.99	2437.38	72515	1	26	413	**6219**	6224.10	0.08	2.33
n30q10E	6001.66	6190.58	32.09	3600.00	58855	11	24	479	6387	6387.00	3.17	1.73
n30q10F	5758.64	5865.48	14.18	3600.00	157820	342	39	388	5977	5977.00	1.90	2.05
n30q10G	8809.99	8946.61	31.72	3600.00	24238	0	29	432	9109	9109.90	1.83	2.66
n30q10H	5822.23	6041.84	18.96	3600.00	40023	3	52	332	6138	6155.30	1.88	2.25
n30q10I	5352.87	5569.63	22.24	3600.00	45502	7	49	258	5764	5764.00	3.49	1.85
n30q10J	5811.77	**6026.00**	22.28	616.94	6908	0	48	315	**6026**	6026.00	0.00	2.10
n30q15A	5402.51	5558.25	16.21	3600.00	59889	7	37	249	5858	5858.00	5.39	1.86
n30q15B	5444.10	5539.50	15.19	3600.00	123154	3218	30	178	5682	5682.00	2.57	1.26
n30q15C	5209.21	5419.08	13.97	3600.00	187041	0	20	144	5636	5636.00	4.00	1.63
n30q15D	5130.73	5274.89	18.60	3600.00	81716	0	39	286	5844	5844.00	10.79	1.56
n30q15E	5273.52	5501.20	16.73	3600.00	155931	6	37	196	5803	5803.00	5.49	1.55
n30q15F	4927.94	5159.51	8.54	3600.00	243010	0	50	97	5388	5388.00	4.43	1.79
n30q15G	7155.00	7243.01	23.46	3600.00	63495	7	41	322	7623	7645.40	5.56	1.47
n30q15H	5063.68	**5304.00**	14.80	686.37	23198	0	46	299	**5304**	5304.00	0.00	1.44
n30q15I	4663.39	**4929.00**	15.54	3448.10	182688	3	32	169	**4929**	4929.00	0.00	1.24
n30q15J	5305.59	5533.33	18.49	3600.00	61803	5	29	268	5847	5847.00	5.67	1.47
n30q20A	4926.84	**5151.00**	9.23	383.54	14858	0	39	102	**5151**	5151.00	0.00	1.21
n30q20B	4922.00	**5336.00**	12.37	2807.90	148741	0	30	181	**5336**	5336.00	0.00	1.09
n30q20C	4943.59	5199.06	9.71	3600.00	178457	0	50	109	5283	5283.00	1.61	1.59
n30q20D	4917.11	**5136.00**	11.30	209.33	2026	0	34	215	**5136**	5136.00	0.00	1.23
n30q20E	4814.96	5041.68	12.79	3600.00	89559	0	21	92	5558	5558.00	10.24	1.25
n30q20F	4667.49	4973.89	8.90	3600.00	229930	0	55	71	5252	5252.00	5.59	1.36
n30q20G	6449.53	6642.42	15.98	3600.00	58747	0	47	236	6821	6821.00	2.69	1.94
n30q20H	4389.43	**4404.00**	8.65	9.45	3	0	7	143	**4404**	4404.00	0.00	0.97
n30q20I	4439.41	4749.42	11.82	3600.00	191420	1	46	110	4868	4868.00	2.50	1.18
n30q20J	4788.99	5027.06	18.08	3600.00	70796	0	32	251	5603	5603.00	11.46	1.64

Table 5: Summary of computational results for $|V| = 20$, $K = 3$ and $N = 2$

Instance	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n20q10A	4913.95	**5226.00**	2.51	256.89	26796	0	22	172	**5226**	5226.00	0.00	0.54
n20q10B	5182.23	5699.89	1.89	3600.00	686323	16	35	66	5920	5920.00	3.86	0.72
n20q10C	6205.47	6444.43	6.58	3600.00	88653	2	23	233	6777	6777.00	5.16	1.07
n20q10D	6235.60	**6621.00**	5.47	2000.34	130466	0	13	200	**6621**	6621.00	0.00	0.94
n20q10E	6437.85	6570.13	4.95	3600.00	137788	14	31	256	6785	6785.00	3.27	0.87
n20q10F	4912.43	5317.50	3.87	3600.00	193083	3	37	147	5788	5788.00	8.85	0.95
n20q10G	5533.56	**5893.00**	3.50	534.57	53601	143	22	151	**5893**	5893.00	0.00	0.80
n20q10H	5713.79	6114.97	3.31	3600.00	190460	3	32	173	7007	7007.00	14.59	0.85

Continued on the next page

Table 5: Results for $|V| = 20$, $K = 3$ and $N = 2$ (continued)

Instance	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n20q10I	4889.26	5174.08	4.41	3600.00	164931	3	18	174	5392	5392.00	4.21	0.87
n20q10J	4452.33	4883.78	2.40	3600.00	347984	23	20	134	5402	5402.00	10.61	0.66
n20q15A	4258.44	4605.96	2.16	3600.00	322579	0	30	57	5035	5035.00	9.31	0.52
n20q15B	4761.68	**5231.00**	2.27	82.19	10625	0	19	105	**5231**	5231.00	0.00	0.64
n20q15C	5431.33	5772.53	4.62	3600.00	127403	0	13	77	6065	6065.00	5.07	0.85
n20q15D	5597.62	5753.91	3.66	3600.00	190030	9	16	140	6258	6258.00	8.76	1.08
n20q15E	5731.41	6016.61	2.56	3600.00	302773	260	19	171	6193	6193.00	2.93	0.61
n20q15F	4739.88	5130.54	3.28	3600.00	211816	0	25	63	5660	5660.00	10.32	0.99
n20q15G	5118.50	**5717.00**	2.60	680.64	71218	44	31	100	**5717**	5717.00	0.00	0.66
n20q15H	4943.23	5649.15	2.87	3600.00	238994	15	45	156	6629	6629.00	17.35	0.80
n20q15I	4388.83	4804.76	2.78	3600.00	345419	0	31	92	5028	5028.00	4.65	0.72
n20q15J	4196.93	4862.54	1.98	3600.00	289317	28	20	100	5300	5300.00	9.00	0.61
n20q20A	3984.79	4431.73	2.29	3600.00	301123	0	28	58	5035	5035.00	13.61	0.51
n20q20B	4736.07	**5231.00**	1.96	123.50	16532	0	18	51	**5231**	5231.00	0.00	0.62
n20q20C	4828.81	5207.42	4.56	3600.00	92333	0	26	142	5926	5926.00	13.80	0.91
n20q20D	4810.42	5105.20	2.85	3600.00	201691	0	20	105	5674	5674.00	11.14	0.96
n20q20E	4889.13	5190.25	2.30	3600.00	372347	0	20	78	5988	5988.00	15.37	0.79
n20q20F	4611.30	4982.10	3.94	3600.00	288753	0	32	50	5404	5404.00	8.47	0.83
n20q20G	4929.38	5487.74	2.63	3600.00	495900	4	23	62	5717	5717.00	4.18	0.63
n20q20H	4624.64	5534.09	2.39	3600.00	307986	0	58	63	6166	6166.00	11.42	0.66
n20q20I	4321.53	4896.87	2.40	3600.00	377333	0	34	85	4954	4954.00	1.17	0.68
n20q20J	4061.48	4691.26	2.03	3600.00	417597	5	26	84	5278	5278.00	12.51	0.55

Table 6: Summary of computational results for $|V| = 30$, $K = 3$ and $N = 2$

Instance	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n30q10A	6374.75	6461.76	20.14	3600.00	38517	0	23	207	6875	6875.00	6.40	2.05
n30q10B	6660.00	6942.05	27.70	3600.00	49464	4	28	286	6972	6972.00	0.43	1.82
n30q10C	6356.29	6541.20	19.87	3600.00	35328	0	20	285	7108	7127.70	8.97	2.08
n30q10D	6075.06	6281.86	20.40	3600.00	31477	0	31	363	6525	6525.00	3.87	2.36
n30q10E	6240.89	6393.62	18.20	3600.00	27387	0	18	296	7122	7122.00	11.39	1.70
n30q10F	5892.82	5998.54	17.80	3600.00	55197	0	34	186	6506	6506.00	8.46	1.50
n30q10G	8876.29	8981.61	21.88	3600.00	32480	0	32	294	9434	9434.00	5.04	2.08
n30q10H	6039.06	6136.15	21.99	3600.00	47352	0	29	316	6698	6801.80	10.85	1.76
n30q10I	5471.99	5599.48	23.04	3600.00	24372	0	21	246	6056	6057.60	8.18	1.47
n30q10J	5939.55	6124.72	21.53	3600.00	32050	0	39	296	6538	6538.00	6.75	2.05
n30q15A	5473.57	5630.85	17.48	3600.00	39488	0	44	132	6243	6243.00	10.87	1.67
n30q15B	5577.28	5772.76	11.24	3600.00	87701	0	25	163	6110	6110.00	5.84	1.55
n30q15C	5365.32	5576.62	13.42	3600.00	49552	0	23	128	6236	6236.00	11.82	1.75
n30q15D	5238.02	5555.68	13.57	3600.00	59961	0	39	200	5798	5798.00	4.36	1.68
n30q15E	5576.55	5748.26	16.10	3600.00	46442	0	16	154	6337	6337.00	10.24	1.52
n30q15F	5210.72	5521.32	10.46	3600.00	113494	0	28	91	5915	5915.00	7.13	1.33

Continued on the next page

Table 6: Results for $|V| = 30$, $K = 3$ and $N = 2$ (continued)

Instance	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
n30q15G	7324.59	7446.31	24.81	3600.00	100272	2	35	278	7600	7600.00	2.06	1.56
n30q15H	5303.88	5462.84	16.90	3600.00	79035	0	21	234	6102	6102.00	11.70	1.96
n30q15I	4772.09	5012.75	14.45	3600.00	37075	0	25	142	5526	5526.00	10.24	1.62
n30q15J	5469.85	5648.32	17.29	3600.00	72059	0	29	172	6034	6034.00	6.83	1.49
n30q20A	5152.29	5473.60	15.00	3600.00	130392	0	33	54	5769	5769.00	5.40	1.30
n30q20B	5137.61	5603.75	13.96	3600.00	80686	0	25	157	6086	6086.00	8.61	1.51
n30q20C	5182.14	5436.03	10.52	3600.00	83120	0	26	92	5731	5731.00	5.43	1.52
n30q20D	5153.15	5473.99	11.14	3600.00	67297	0	32	90	5872	5872.00	7.27	1.51
n30q20E	5064.51	5272.76	14.39	3600.00	64516	0	20	88	6014	6014.00	14.06	1.67
n30q20F	4973.41	5371.83	8.84	3600.00	171851	0	39	53	5601	5601.00	4.27	1.09
n30q20G	6751.61	**6968.00**	16.88	461.85	25920	14	33	119	**6968**	6968.00	0.00	1.15
n30q20H	4538.24	4657.55	9.66	3600.00	104608	0	26	103	5679	5679.00	21.93	1.79
n30q20I	4598.61	4788.42	12.16	3600.00	57358	0	28	87	5410	5410.00	12.98	1.26
n30q20J	4876.91	5224.61	14.20	3600.00	56209	0	25	141	5980	6000.40	14.85	1.38

For almost all cases, BC could not be run to completion within the time limit of one hour. Yet, the exact algorithm managed to solve 38 instances to optimality: 23 for $|V| = 20$ and $K = 2$, 8 for $|V| = 30$ and $K = 2$, 6 for $|V| = 20$ and $K = 3$, and 1 for $|V| = 30$ and $K = 3$. Note that ILS also found the optimal solutions for these 38 instances, but within three seconds. As expected, the root time for $|V| = 30$ is larger than $|V| = 20$ because the CPU time required to solve the linear programs naturally increases with the size of the instance. Since this also happens in the other nodes, a smaller number of nodes could be solved within the time limit for the instances involving 30 vertices, when compared to those containing 20 vertices. ILS found always the same solution for the 10 runs, except for instances n30q10H, n30q15G, n30q10C, n30q10D, n30q10I, n30q20J, thus suggesting that the proposed heuristic has a consistent performance in terms of robustness. The algorithm also runs very fast for these instances, never spending, on average, more than three seconds (except for instance n30q10C). Detailed results for all instances with up to 200 vertices can be found in appendix (see supplementary material).

5.5. Aggregate results

Tables 7 and 8 show the aggregate results for $K = 2$ and $K = 3$, respectively, where $\mathbf{Gap_{LB}}$ denotes the average gap between the average solutions and the lower bound found by BC, $\mathbf{Gap_{BKS}}$ represents the average gap between the average solutions and the best known solutions (BKSs) and **Time (s)** corresponds to the average CPU time in seconds.

We do not report the gaps with respect to the lower bound for the instances involving 200 vertices, because BC failed in most cases to solve the linear relaxation within the time limit.

Table 7: Aggregate results for $K = 2$

$\|V\|$	$Q = 10$			$Q = 15$			$Q = 20$		
	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)
20	2.26	0.00	0.75	0.48	0.00	0.64	1.44	0.00	0.59
30	2.09	0.04	2.06	4.39	0.03	1.53	3.41	0.00	1.35
40	4.52	0.33	3.37	5.82	0.04	2.98	6.24	0.01	2.78
50	7.40	0.60	7.45	6.19	0.17	5.76	4.81	0.07	4.34
60	11.53	1.21	11.71	10.10	0.47	7.89	7.98	0.22	7.57
100	20.46	2.41	40.26	17.66	1.73	37.60	14.32	1.04	31.16
200	-	1.58	293.32	-	2.29	235.80	-	2.04	300.37

Table 8: Aggregate results for $K = 3$

$\|V\|$	$Q = 10$			$Q = 15$			$Q = 20$		
	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)
20	5.06	0.00	0.83	6.74	0.00	0.75	9.17	0.00	0.71
30	7.03	0.19	1.89	8.11	0.00	1.61	9.48	0.03	1.42
40	9.39	0.46	3.55	10.21	0.11	2.91	11.13	0.00	2.87
50	9.55	0.72	7.57	8.96	0.27	5.23	9.43	0.08	4.82
60	14.63	1.67	10.90	12.55	0.40	8.90	12.38	0.22	8.13
100	21.38	2.26	45.95	19.80	1.73	36.46	16.53	1.36	31.96
200	-	2.35	334.18	-	2.32	260.28	-	2.00	293.20

The results demonstrate that there is a considerable gap between the average solution values and the lower bound. This does not necessarily mean that the solutions generated by ILS are of poor quality. In fact, based on the small values of the average gaps with respect to the BKSs, we suspect that the upper bounds found by ILS is likely to be closer to the optimal solution than the lower bounds obtained by BC. For example, for the instances involving up to 50 vertices, it can be observed that the average gaps with respect to the BKSs are always smaller than 1%, which suggests that ILS systematically finds, on average, potentially high quality solutions. Furthermore, the average CPU time of the proposed heuristic can be considered acceptable, especially for the instances containing up to 100 vertices.

It is important to point ot that we have disregarded the runs for which ILS was unable to generate a feasible solution. Tables 9 and 10 illustrate those instances for $K = 2$ and $K = 3$, respectively, where the proposed heuristic failed to find a feasible solution in at least one of the 10 runs.

There are relatively very few instances for which ILS was not successful in finding 10 feasible solutions out of 10 runs. This happened in 20 instances for $K = 2$ (9.52%) and in

Table 9: Instances with less than 10 feasible runs for $K = 2$

Number of feasible runs and corresponding instances								
1	2	3	4	5	6	7	8	9
-	n200q10D	-	n200q10C	n200q15F	n100q10B	n50q20D	n100q15C	n50q15I
	n200q10F		n200q15A			n200q15E	n100q20F	n100q10D
	n200q10H						n200q10A	n100q20E
							n200q10J	n200q10B
							n200q15C	
							n200q15I	
							n200q20E	

Table 10: Instances with less than 10 feasible runs for $K = 3$

Number of feasible runs and corresponding instances								
1	2	3	4	5	6	7	8	9
-	n200q10B	n200q10F	n200q15A	-	n200q10E	-	n100q10D	n40q10D
		n200q10J	n200q15D		n200q10H		n200q10A	n40q15H
			n200q15F		n200q15B		n200q10C	n50q15I
								n60q10B
								n100q10F
								n100q15A
								n200q10G
								n200q15C
								n200q20F

21 instances for $K = 3$ (10.00%). We can also observe that, in most cases, these instances contain 200 vertices. These results suggest that, depending on the value of L, finding feasible SBRP-MVV solutions in a systematic fashion is a challenging task.

5.6. Results for the real-world instances of Rainer-Harbach et al. (2015)

We now report results for the real-world instances of Rainer-Harbach et al. (2015).

5.6.1. Detailed results for small size instances

Tables 11-13 present the detailed results for the small size real-world instances of Rainer-Harbach et al. (2015), in particular, those with up to 30 vertices. All 10-vertex instances were solved within less than a second, while 28 and 21 instances involving 20 and 30 vertices were solved to optimality, respectively. Moreover, note that the lazy cuts were never required for such instances. Finally, all known optimal solutions were found by ILS, and the objective value of the average solutions were, on average, very close to the one associated with the best known solutions.

Table 11: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 10$

	BC								ILS			
Instance-(Q,K)	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)

Continued on the next page

Table 11: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 10$ (continued)

Instance-(Q,K)	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
B_010_1_00-(20,1)	281.00	**281.00**	0.09	0.09	1	0	1	35	**281**	281.00	0.00	0.09
B_010_1_01-(20,1)	303.55	**305.00**	0.27	0.36	8	0	6	56	**305**	305.00	0.00	0.10
B_010_1_02-(20,1)	224.00	**224.00**	0.19	0.19	1	0	4	44	**224**	224.00	0.00	0.08
B_010_1_03-(20,1)	265.00	**265.00**	0.10	0.10	1	0	2	26	**265**	265.00	0.00	0.14
B_010_1_04-(20,1)	289.00	**289.00**	0.34	0.54	29	0	12	50	**289**	289.00	0.00	0.12
B_010_1_05-(20,1)	276.00	**276.00**	0.26	0.26	1	0	4	28	**276**	276.00	0.00	0.09
B_010_1_06-(20,1)	256.00	**256.00**	0.07	0.07	1	0	1	0	**256**	256.00	0.00	0.06
B_010_1_07-(20,1)	259.02	**260.00**	0.33	0.43	11	0	3	60	**260**	260.00	0.00	0.08
B_010_1_08-(20,1)	258.00	**258.00**	0.15	0.15	1	0	1	42	**258**	258.00	0.00	0.13
B_010_1_09-(20,1)	268.00	**268.00**	0.10	0.10	1	0	0	42	**268**	268.00	0.00	0.09
B_010_1_10-(20,1)	219.00	**221.00**	0.20	0.29	14	0	11	152	**221**	221.00	0.00	0.13
B_010_1_11-(20,1)	245.00	**245.00**	0.25	0.25	1	0	4	101	**245**	245.00	0.00	0.10
B_010_1_12-(20,1)	266.00	**266.00**	0.09	0.09	1	0	2	38	**266**	266.00	0.00	0.11
B_010_1_13-(20,1)	235.00	**235.00**	0.07	0.07	1	0	0	44	**235**	235.00	0.00	0.07
B_010_1_14-(20,1)	273.00	**273.00**	0.25	0.25	1	0	1	110	**273**	273.00	0.00	0.09
B_010_1_15-(20,1)	251.00	**251.00**	0.25	0.25	1	0	5	98	**251**	251.00	0.00	0.14
B_010_1_16-(20,1)	256.00	**256.00**	0.39	0.55	31	0	10	174	**256**	256.00	0.00	0.14
B_010_1_17-(20,1)	290.66	**291.00**	0.29	0.35	5	0	2	83	**291**	291.00	0.00	0.14
B_010_1_18-(20,1)	277.00	**277.00**	0.08	0.08	1	0	1	42	**277**	277.00	0.00	0.10
B_010_1_19-(20,1)	242.00	**242.00**	0.28	0.28	1	0	2	59	**242**	242.00	0.00	0.11
B_010_1_20-(20,1)	208.00	**208.00**	0.05	0.05	1	0	2	42	**208**	208.00	0.00	0.07
B_010_1_21-(20,1)	255.00	**255.00**	0.10	0.10	1	0	2	54	**255**	255.00	0.00	0.09
B_010_1_22-(20,1)	245.00	**245.00**	0.06	0.06	1	0	1	38	**245**	245.00	0.00	0.09
B_010_1_23-(20,1)	258.00	**258.00**	0.23	0.23	1	0	3	54	**258**	258.00	0.00	0.08
B_010_1_24-(20,1)	209.92	**212.00**	0.29	0.48	21	0	10	80	**212**	212.00	0.00	0.08
B_010_1_25-(20,1)	255.00	**255.00**	0.09	0.09	1	0	2	34	**255**	255.00	0.00	0.08
B_010_1_26-(20,1)	252.00	**252.00**	0.09	0.09	1	0	2	64	**252**	252.00	0.00	0.11
B_010_1_27-(20,1)	210.00	**210.00**	0.09	0.09	1	0	1	28	**210**	210.00	0.00	0.06
B_010_1_28-(20,1)	249.74	**251.00**	0.22	0.48	34	0	14	61	**251**	251.00	0.00	0.12
B_010_1_29-(20,1)	323.00	**324.00**	0.17	0.30	23	0	5	47	**324**	324.00	0.00	0.09

Table 12: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 20$

Instance-(Q,K)	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
B_020_1_00-(20,1)	440.60	**442.00**	1.55	5.72	60	0	9	134	**442**	442.00	0.00	0.38
B_020_1_01-(20,1)	465.60	**469.00**	1.75	19.20	655	0	44	88	**469**	469.00	0.00	0.58
B_020_1_02-(20,2)	489.52	500.59	2.29	3600.00	480532	0	131	144	517	517.00	3.28	0.81
B_020_1_03-(20,1)	355.96	**357.00**	1.05	2.51	20	0	9	177	**357**	357.00	0.00	0.27
B_020_1_04-(20,2)	558.77	**581.00**	2.52	1862.48	374563	0	102	87	**581**	581.00	0.00	0.84
B_020_1_05-(20,1)	467.00	**467.00**	0.49	0.49	1	0	0	31	**467**	467.60	0.13	0.33
B_020_1_06-(20,1)	466.50	**468.00**	2.18	11.71	265	0	31	149	**468**	468.00	0.00	0.45

Continued on the next page

Table 12: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 20$ (continued)

Instance-(Q,K)	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
B_020_1_07-(20,2)	513.47	**532.00**	3.13	519.67	62675	0	86	97	**532**	532.00	0.00	0.91
B_020_1_08-(20,1)	404.28	**406.00**	1.51	3.74	30	0	5	122	**406**	406.00	0.00	0.26
B_020_1_09-(20,1)	477.00	**477.00**	1.71	1.71	1	0	5	153	**477**	477.00	0.00	0.40
B_020_1_10-(20,1)	467.19	**468.00**	2.07	3.41	13	0	1	111	**468**	468.00	0.00	0.49
B_020_1_11-(20,1)	473.91	**476.00**	2.13	8.09	219	0	22	136	**476**	476.00	0.00	0.40
B_020_1_12-(20,2)	527.72	**536.00**	2.53	30.67	2015	0	21	154	**536**	536.00	0.00	0.88
B_020_1_13-(20,2)	478.99	**507.00**	2.68	2132.47	146374	0	170	136	**507**	507.20	0.04	0.95
B_020_1_14-(20,2)	528.41	**563.00**	2.26	739.38	60715	0	137	125	**563**	563.00	0.00	0.82
B_020_1_15-(20,2)	489.74	**527.00**	2.30	600.69	27106	0	71	135	**527**	527.00	0.00	0.93
B_020_1_16-(20,1)	436.93	**437.00**	1.87	3.03	11	0	9	152	**437**	437.00	0.00	0.40
B_020_1_17-(20,1)	444.84	**447.00**	1.71	9.01	157	0	19	270	**447**	447.00	0.00	0.47
B_020_1_18-(20,1)	454.46	**456.00**	1.22	9.77	209	0	20	53	**456**	456.00	0.00	0.53
B_020_1_19-(20,1)	417.89	**421.00**	1.70	13.56	328	0	25	161	**421**	421.00	0.00	0.48
B_020_1_20-(20,2)	518.52	**537.00**	1.84	49.59	1218	0	70	54	**537**	537.00	0.00	0.71
B_020_1_21-(20,2)	498.70	**527.00**	3.25	753.24	30974	0	70	146	**527**	527.00	0.00	0.87
B_020_1_22-(20,1)	470.00	**470.00**	1.40	1.40	1	0	3	102	**470**	470.00	0.00	0.39
B_020_1_23-(20,2)	505.24	**538.00**	3.28	2445.35	90682	0	150	103	**538**	538.00	0.00	1.06
B_020_1_24-(20,2)	486.56	**514.00**	2.70	458.59	21206	0	158	112	**514**	514.00	0.00	1.01
B_020_1_25-(20,2)	517.14	**524.00**	2.85	80.47	9442	0	54	72	**524**	524.00	0.00	0.87
B_020_1_26-(20,2)	498.62	**521.00**	3.42	819.21	68586	0	132	136	**521**	521.00	0.00	0.86
B_020_1_27-(20,2)	497.31	502.61	1.52	3600.00	995200	0	195	67	518	518.00	3.06	0.68
B_020_1_28-(20,2)	535.81	**556.00**	2.25	97.99	6452	0	68	110	**556**	556.00	0.00	0.86
B_020_1_29-(20,1)	427.62	**431.00**	1.36	3.46	71	0	6	63	**431**	431.00	0.00	0.39

Table 13: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 30$

Instance-(Q,K)	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
B_030_1_00-(20,2)	759.04	**771.00**	16.56	2037.43	71315	0	37	152	**771**	771.80	0.10	2.77
B_030_1_01-(20,2)	703.41	709.40	15.63	3600.00	62744	0	74	67	725	725.40	2.26	2.15
B_030_1_02-(20,2)	736.47	747.24	12.59	3600.00	33783	0	60	76	770	770.50	3.11	2.01
B_030_1_03-(20,2)	781.09	**786.00**	9.25	3174.96	250005	0	54	154	**786**	787.30	0.17	2.36
B_030_1_04-(20,2)	734.08	**738.00**	8.09	192.34	1846	0	39	83	**738**	738.00	0.00	1.80
B_030_1_05-(20,2)	769.78	**784.00**	8.60	3212.58	148955	0	83	115	**784**	785.10	0.14	2.23
B_030_1_06-(20,2)	758.85	766.07	13.67	3600.00	123486	0	150	91	780	780.00	1.82	2.17
B_030_1_07-(20,2)	792.13	798.01	13.92	3600.00	156655	0	39	130	808	808.40	1.30	2.34
B_030_1_08-(20,2)	714.24	**728.00**	14.58	589.36	23151	0	42	190	**728**	728.00	0.00	1.72
B_030_1_09-(20,2)	768.64	**784.00**	7.54	521.50	26570	0	64	61	**784**	784.40	0.05	1.75
B_030_1_10-(20,2)	701.86	**723.00**	11.51	1765.46	50848	0	68	149	**723**	723.20	0.03	1.68
B_030_1_11-(20,2)	780.50	**788.00**	12.29	398.14	35297	0	21	236	**788**	788.10	0.01	2.11
B_030_1_12-(20,2)	779.00	786.10	12.60	3600.00	201519	0	92	91	792	792.10	0.76	2.29
B_030_1_13-(20,2)	700.81	**716.00**	8.34	1607.65	72754	0	67	98	**716**	716.50	0.07	1.87

Continued on the next page

Table 13: Results for the instances of Rainer-Harbach et al. (2015) with $|V| = 30$ (continued)

Instance-(Q,K)	BC								ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	#Lazy Cuts	#Subtour Cuts	#Cap. Cuts	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
B_030_1_14-(20,2)	734.31	752.08	13.59	3600.00	130684	0	48	219	753	754.00	0.26	1.95
B_030_1_15-(20,2)	676.00	**696.00**	9.42	3341.72	188744	0	79	103	**696**	696.50	0.07	1.37
B_030_1_16-(20,2)	714.47	**727.00**	9.22	554.09	13708	0	58	115	**727**	727.00	0.00	1.40
B_030_1_17-(20,2)	742.88	**749.00**	9.85	1400.42	67025	0	57	168	**749**	749.90	0.12	2.39
B_030_1_18-(20,2)	597.89	**601.00**	8.14	321.08	4969	0	59	165	**601**	601.00	0.00	1.34
B_030_1_19-(20,2)	723.09	728.62	9.54	3600.00	135000	0	62	184	742	742.00	1.84	1.65
B_030_1_20-(20,2)	833.73	**837.00**	10.56	1104.01	127310	0	66	212	838	838.40	0.17	2.08
B_030_1_21-(20,2)	718.15	**731.00**	11.22	958.47	52259	0	45	204	**731**	732.00	0.14	2.10
B_030_1_22-(20,2)	751.78	766.38	12.74	3600.00	137593	0	68	210	769	769.50	0.41	2.21
B_030_1_23-(20,2)	646.59	**659.00**	10.95	378.99	18385	0	26	75	**659**	659.20	0.03	2.14
B_030_1_24-(20,2)	784.70	798.42	10.59	3600.00	105128	0	48	111	804	804.40	0.75	2.34
B_030_1_25-(20,2)	784.64	**789.00**	11.33	755.38	26473	0	67	253	**789**	789.90	0.11	2.60
B_030_1_26-(20,2)	677.18	**695.00**	12.54	1061.05	24782	0	72	175	**695**	695.50	0.07	1.91
B_030_1_27-(20,2)	702.41	**721.00**	9.49	2647.75	212500	0	147	138	**721**	721.50	0.07	1.62
B_030_1_28-(20,2)	774.11	**789.00**	8.40	646.12	20360	0	81	82	**789**	789.00	0.00	2.32
B_030_1_29-(20,2)	677.31	**691.00**	7.37	752.51	39548	0	47	216	**691**	691.00	0.00	1.57

5.6.2. Aggregate results

Table 14 reports the aggregate results for all groups of instances, while the results for each instance are reported in appendix (see supplementary material). The small values of Gap_{BKS} suggest that ILS has a robust performance at least in terms of finding solutions with similar objective values over the 10 runs. In addition, feasible solutions were found for all runs. Furthermore, it is interesting to observe that the average gaps with respect to the lower bound is considerably smaller to those found reported in Section 5.5. Given the consistent performance of ILS, we believe that the lower bounds for this set of instances are of better quality than those obtained for the previous set.

Table 14: Aggregate results for the instances of Rainer-Harbach et al. (2015)

$\|V\|$	Gap_{LB} (%)	Gap_{BKS} (%)	Time (s)
10	0.00	0.00	0.10
20	0.22	0.01	0.64
30	0.46	0.06	2.01
60	2.96	0.24	10.75
90	3.26	0.30	33.09
120	3.64	0.27	73.63
180	4.50	0.33	297.07

5.7. Results for the real-world instances of Dell'Amico et al. (2014)

Table 15 shows the results obtained for the real-world based instances derived from Dell'Amico et al. (2014). In this case we do not report the number of cuts due to space restrictions. We can observe that 22 optimal solutions were found and that all of them were quickly identified by ILS. As reported in the appendix (see supplementary material), ILS found feasible solutions in all runs for this set of instances.

Table 15: Results for the real-world instances of Dell'Amico et al. (2014)

	BC					ILS			
Instance-($\|V\|, Q, K$)	Root LB	LB	Root Time (s)	Time (s)	Tree Size	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
3Bari10-(13,10,2)	20600.00	**20600.00**	0.20	0.20	1	**20600**	20600.00	0.00	0.27
2Bari20-(13,20,2)	16299.12	**19300.00**	0.21	78.05	46730	**19300**	19300.00	0.00	0.24
1Bari30-(13,30,2)	15747.03	**19300.00**	0.36	103.34	56076	**19300**	19300.00	0.00	0.24
6ReggioEmilia10-(14,10,3)	32561.49	**34100.00**	0.94	1030.80	302333	**34100**	34100.00	0.00	0.47
5ReggioEmilia20-(14,20,2)	23509.02	**24400.00**	0.76	3.74	558	**24400**	24400.00	0.00	0.33
4ReggioEmilia30-(14,30,2)	21450.25	**24300.00**	0.75	6.72	817	**24300**	24300.00	0.00	0.33
12Parma10-(15,10,2)	32477.55	**33600.00**	0.70	3.76	351	**33600**	33600.00	0.00	0.36
9Bergamo12-(15,12,2)	13888.74	**14400.00**	1.09	4.94	565	**14400**	14400.00	0.00	0.48
11Parma20-(15,20,2)	30764.76	**32900.00**	0.45	6.13	682	**32900**	32900.00	0.00	0.31
8Bergamo20-(15,20,2)	13313.38	**13500.00**	0.80	1.02	11	**13500**	13500.00	0.00	0.36
10Parma30-(15,30,2)	30764.76	**32900.00**	0.50	7.43	765	**32900**	32900.00	0.00	0.31
7Bergamo30-(15,30,2)	13170.68	**14100.00**	0.66	12.24	1769	**14100**	14100.00	0.00	0.31
15Treviso10-(18,10,2)	31322.26	**33905.00**	1.44	1088.05	319798	**33905**	33905.00	0.00	0.41
14Treviso20-(18,20,2)	31065.18	**33905.00**	1.82	689.17	199562	**33905**	33905.00	0.00	0.41
13Treviso30-(18,30,2)	31057.89	**33905.00**	1.33	493.05	136433	**33905**	33905.00	0.00	0.41
18LaSpezia10-(20,10,2)	23666.33	**26571.00**	2.48	2829.61	190910	**26571**	26571.00	0.00	0.81
17LaSpezia20-(20,20,2)	21339.41	23124.36	2.02	3600.00	284776	26539	26539.00	14.77	0.64
16LaSpezia30-(20,30,2)	21312.26	22907.88	1.74	3600.00	380739	26539	26539.00	15.85	0.65
23Ottawa10-(21,10,2)	19587.76	22326.85	2.86	3600.00	348490	23499	23499.00	5.25	0.66
20BuenosAires20-(21,20,2)	86430.34	88229.15	8.46	3600.00	164163	90303	90669.40	2.77	3.45
22Ottawa20-(21,20,2)	17430.98	21469.07	1.75	3600.00	478702	23499	23499.00	9.46	0.67
19BuenosAires30-(21,30,2)	73135.63	75822.68	12.33	3600.00	91308	80222	80611.80	6.32	2.65
21Ottawa30-(21,30,2)	17435.77	20723.84	1.71	3600.00	450511	23499	23499.00	13.39	0.69
26SanAntonio10-(23,10,4)	39553.00	39661.00	5.79	3600.00	227366	40535	40535.50	2.20	1.80
25SanAntonio20-(23,20,2)	23541.21	**24654.00**	2.92	203.62	30549	**24654**	24654.00	0.00	0.85
24SanAntonio30-(23,30,2)	23315.00	**23315.00**	3.57	3.83	3	**23315**	23315.00	0.00	0.65
29Brescia11-(27,11,4)	37783.68	38473.13	10.44	3600.00	103260	39900	40150.00	4.36	2.43
28Brescia20-(27,20,3)	33539.74	**35200.00**	8.34	1577.36	141369	**35200**	35200.00	0.00	1.52
27Brescia30-(27,30,2)	31900.00	**31900.00**	2.64	2.64	1	**31900**	31900.00	0.00	1.08
35Madison10-(28,10,2)	35594.58	38956.20	10.44	3600.00	81776	42089	42089.00	8.04	1.49
32Roma18-(28,18,6)	66694.75	67171.75	20.68	3600.00	56624	98000	98360.00	46.43	4.06
31Roma20-(28,20,6)	65207.54	65554.96	14.82	3600.00	63855	95600	95630.00	45.88	3.87
34Madison20-(28,20,2)	32343.82	36109.87	9.06	3600.00	115903	39895	39895.00	10.48	1.40
30Roma30-(28,30,4)	61401.06	63016.22	18.20	3600.00	26141	91900	91900.00	45.84	3.07
33Madison30-(28,30,2)	31182.34	35793.35	5.19	3600.00	112274	38947	38947.00	8.81	1.35
38Guadalajara11-(41,11,4)	63136.15	66348.81	19.97	3600.00	148309	67892	67892.00	2.33	4.41
37Guadalajara20-(41,20,3)	57691.64	**61543.00**	21.89	482.09	23571	**61543**	61543.00	0.00	3.34

Continued on the next page

Table 15: Results for the real-world instances of Dell'Amico et al. (2014) (continued)

Instance-($\|V\|, Q, K$)	BC					ILS			
	Root LB	LB	Root Time (s)	Time (s)	Tree Size	Min Cost	Avg Cost	Avg Gap (%)	Avg Time (s)
36Guadalajara30-(41,30,2)	57265.75	**58136.00**	23.80	162.78	1146	**58136**	58140.60	0.01	3.20
41Dublin11-(45,11,6)	51837.04	52083.38	144.18	3600.00	4619	54622	55040.80	5.68	7.79
40Dublin20-(45,20,4)	38136.08	38588.32	79.33	3600.00	5228	41123	41156.70	6.66	5.50
39Dublin30-(45,30,3)	32944.07	33760.99	52.70	3600.00	17842	34652	34652.00	2.64	4.96
44Denver10-(51,10,5)	59503.90	60361.63	127.93	3600.00	11216	86003	86701.80	43.64	12.19
43Denver20-(51,20,2)	49972.72	51749.51	99.78	3600.00	6934	54691	54728.00	5.76	6.53
42Denver30-(51,30,2)	48594.43	51217.64	76.99	3600.00	13911	53421	53421.00	4.30	6.31
47RioDeJaneiro10-(55,10,9)	248440.47	248574.69	414.02	3600.00	1834	264880	266276.30	7.12	15.11
46RioDeJaneiro20-(55,20,5)	153851.32	154298.35	267.79	3600.00	1910	160651	162097.80	5.05	9.68
45RioDeJaneiro30-(55,30,3)	121726.14	121971.65	139.04	3600.00	2515	126157	126700.20	3.88	9.77
50Boston16-(59,16,2)	73101.72	74188.37	245.68	3600.00	1268	78460	79150.60	6.69	18.06
49Boston20-(59,20,2)	69363.23	70875.31	216.50	3600.00	2835	77016	77377.90	9.17	14.50
48Boston30-(59,30,2)	68802.74	70163.90	103.03	3600.00	2472	75101	75195.00	7.17	12.19
53Torino10-(75,10,4)	57000.79	57005.10	651.60	3600.00	576	64215	65979.60	15.74	30.03
52Torino20-(75,20,2)	47908.02	48106.69	428.96	3600.00	760	51764	52496.10	9.12	21.34
51Torino30-(75,30,2)	46972.81	47313.51	248.08	3600.00	1117	48377	48744.30	3.02	18.52
56Toronto12-(80,12,2)	51747.49	51760.76	2592.23	3600.00	13	62522	64066.10	23.77	25.27
55Toronto20-(80,20,2)	42438.57	42455.21	1167.45	3600.00	68	50370	51212.40	20.63	33.62
54Toronto30-(80,30,2)	40368.37	40469.65	788.90	3600.00	238	46010	46524.90	14.96	27.12
59Miami10-(82,10,20)	382998.58	383005.11	1903.11	3600.00	138	399037	399748.00	4.37	56.94
58Miami20-(82,20,10)	204259.57	204263.82	1726.72	3600.00	57	215821	217099.50	6.28	38.16
57Miami30-(82,30,7)	144906.60	144960.75	2232.41	3600.00	43	159598	160036.00	10.40	31.75
62CiudadDeMexico17-(90,17,6)	95052.22	95053.51	3492.02	3600.00	2	107313	108592.50	14.24	62.95
61CiudadDeMexico20-(90,20,6)	85165.16	85166.19	3229.30	3600.00	6	102348	103507.40	21.54	60.29
60CiudadDeMexico30-(90,30,3)	64588.86	64610.30	3350.30	3600.00	3	72029	73164.10	13.24	72.06
65Minneapolis10-(116,10,11)	234192.38	234192.38	3600.00	3600.00	1	268584	271430.20	15.90	99.62
64Minneapolis20-(116,20,5)	151993.93	151993.93	3600.00	3600.00	1	174949	176814.50	16.33	77.38
63Minneapolis30-(116,30,4)	131614.25	131756.57	2377.30	3600.00	7	151804	153494.40	16.50	62.21

6. Concluding remarks

We have presented the *static bike relocation problem with multiple vehicles and visits* (SBRP-MVV), and we have proposed a branch-and-cut (BC) algorithm over an extended network-based mathematical formulation. The constraints that ensure that a station is never visited by more than one vehicle were added dynamically, that is, only when an infeasible integer solution was found. In addition, we have also developed an iterated local search (ILS) based heuristic that uses efficient auxiliary data structures to perform move evaluations in amortized constant time during the local search. This was done by storing several information of each subsequence of a current solution.

Extensive computational experiments were conducted on 1325 new benchmark instances ranging from 10 to 200 vertices. The results obtained reveal that multiple visits are only interesting for those instances whose vehicle capacity is up to 20. Moreover, the average gaps between the average solutions found by ILS and the lower bound obtained by BC appear to increase with the number of vehicles. On the other hand, the number of multiple visits is likely to decrease as the number of vehicles increases.

Acknowledgments

This work was partially supported by the Brazilian research agency CNPq, grant 305223/2015-1, and by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged. Thanks are due to the referees for their valuable comments.

References

R. Alvarez-Valdes, J. M. Belenguer, E. Benavent, J. D. Bermudez, F. Muñoz, E. Vercher, F. Verdejo, Optimizing the level of service quality of a bike-sharing system, Omega 62 (2016) 163–175.

P. Augerat, J. Belenguer, E. Benavent, A. Corberán, D. Naddef, Separating capacity constraints in the CVRP using tabu search, European Journal of Operational Research 106 (2) (1998) 546 – 557.

M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, L. Robinet, Balancing the stations of a self-service bike hire system, RAIRO-Operations Research 45 (2011) 37–61.

D. Chemla, F. Meunier, T. Pradeau, R. Wolfler Calvo, H. Yahiaoui, Self-service bike sharing systems: simulation, repositioning, pricing, Tech. Rep. hal-00824078, Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique - CERMICS, Laboratoire d'Informatique de Paris-Nord - LIPN , Parallélisme, Réseaux, Systèmes d'information, Modélisation - PRISM, 2013a.

D. Chemla, F. Meunier, R. Wolfler Calvo, Bike sharing systems: Solving the static rebalancing problem, Discrete Optimization 10 (2013b) 120–146.

C. Contardo, C. Morency, L.-M. Rousseau, Balancing a dynamic public bike-sharing system, Tech. Rep., CIRRELT-2012-09, Montréal, Canada, 2012.

F. Cruz, A. Subramanian, B. P. Bruck, M. Iori, A heuristic algorithm for a single vehicle static bike sharing rebalancing problem, Computers & Operations Research 79 (2017) 19–33.

M. Dell'Amico, E. Hadjicostantinou, M. Iori, S. Novellani, The bike sharing rebalancing problem: Mathematical formulations and benchmark instances, Omega 45 (2014) 7–19.

M. Dell'Amico, M. Iori, S. Novellani, T. Stützle, A destroy and repair algorithm for the bike sharing rebalancing problem, Computers & Operations Research 71 (2016) 149–162.

G. Erdoğan, M. Battarra, R. Wolfler Calvo, An exact algorithm for the static rebalancing problem arising in bicycle sharing systems, European Journal of Operational Research 245 (3) (2015) 667–679.

G. Erdoğan, G. Laporte, R. Wolfler Calvo, The static bicycle relocation problem with demand intervals, European Journal of Operational Research 238 (2014) 451–457.

H. M. Espegren, J. Kristianslund, H. Andersson, K. Fagerholt, The static bicycle repositioning problem - literature survey and new formulation, in: A. Paias, M. Ruthmair, S. Voß (Eds.), Computational Logistics: 7th International Conference, ICCL 2016, Lisbon, Portugal, September 7-9, 2016, Proceedings, Springer International Publishing, Cham, 337–351, 2016.

I. A. Forma, T. Raviv, M. Tzur, A 3-step math heuristic for the static repositioning problem in bike-sharing systems, Transportation Research Part B: Methodological 71 (2015) 230–247.

L. D. Gaspero, A. Rendl, T. Urli, A hybrid ACO+CP for balancing bicycle sharing systems, in: M. Blesa, C. Blum, P. Festa, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, vol. 7919 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 198–212, 2013.

H. Hernández-Pérez, J.-J. Salazar-González, Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, Transportation Science 38 (2) (2004) 245–255.

S. C. Ho, W. Y. Szeto, Solving a static repositioning problem in bike-sharing systems using iterated tabu search, Transportation Research Part E: Logistics and Transportation Review 69 (2014) 180–198.

G. Laporte, F. Meunier, R. Wolfler Calvo, Shared mobility systems, 4OR 13 (4) (2015) 341–360.

Y. Li, W. Y. Szeto, J. Long, C. S. Shui, A multiple type bike repositioning problem, Transportation Research Part B: Methodological 90 (2016) 263–278.

J. H. Lin, T. C. Chou, A geo-aware and VRP-based public bicycle redistribution system, International Journal of Vehicular Technology 2012 (2012) 1–14.

H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search: Framework and applications, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, vol. 146 of *International Series in Operations Research & Management Science*, Springer, New York, 363–397, 2010.

N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (1997) 1097–1100.

R. Nair, E. Miller-Hooks, Fleet management for vehicle sharing operations, Transportation Science 45 (2011) 524–540.

P. H. V. Penna, A. Subramanian, L. S. Ochi, An iterated local search heuristic for the heterogeneous fleet vehicle routing problem, Journal of Heuristics 19 (2013) 201–232.

M. Rainer-Harbach, P. Papazek, G. Raidl, B. Hu, C. Kloimullner, PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems, Journal of Global Optimization 63 (3) (2015) 597–629.

T. Raviv, M. Tzur, I. Forma, Static repositioning in a bike-sharing system: models and solution approaches, EURO Journal on Transportation and Logistics 2 (2013) 187–229.

M. G. Resende, C. C. Ribeiro, Greedy randomized adaptive search procedures: Advances, hybridizations, and applications, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, vol. 146 of *International Series in Operations Research & Management Science*, Springer, New York, 283–319, 2010.

M. M. Silva, A. Subramanian, L. S. Ochi, An iterated local search heuristic for the split delivery vehicle routing problem, Computers & Operations Research 53 (0) (2015) 234–249.

A. Subramanian, Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems, Ph.D. thesis, Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, 2012.

A. Subramanian, L. M. A. Drummond, C. Bentes, L. S. Ochi, R. Farias, A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery, Computers & Operations Research 37 (11) (2010) 1899–1911.

T. Vidal, M. Battarra, A. Subramanian, G. Erdoğan, Hybrid metaheuristics for the clustered vehicle routing problem, Computers & Operations Research. 58 (2015) 87–99.